Laboratorio: Modelos de regresión lineal con R

Beatriz Pateiro López

1 Ajuste de un modelo de regresión lineal con R **2**
1.1 Estimación de los parámetros del modelo 3
1.2 Contrastes sobre los parámetros del modelo 4
1.3 Predicción 5

2 Ajuste de los parámetros de un modelo de regresión lineal mediante métodos de optimización **5**
2.1 Método de descenso de gradiente 6
2.2 Método de descenso de gradiente estocástico (Stochastic Gradient Descent) 7

3 Problemas en el ajuste del modelo de regresión lineal **7**
3.1 Efecto de la multicolinealidad en la estimación del modelo de regresión lineal por el método de mínimos cuadrados 7
3.2 Efecto de la alta dimensión en la estimación del modelo de regresión lineal por el método de mínimos cuadrados 9

4 Regresión lineal con regularización **9**
4.1 Regresión Ridge 9
4.2 Regresión Lasso (least absolute shrinkage and selection operator) 12
4.3 Selección del parámetro de penalización 13

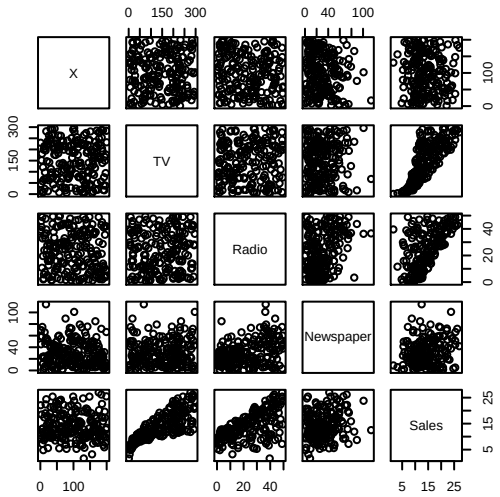
En esta práctica repasaremos lo comandos básicos de R para ajustar un modelo de regresion lineal. Además de la estimación de los parámetros del modelo, realizaremos los contrastes estadísticos oportunos para verificar la validez del modelo propuesto. Utilizaremos como ejemplo los datos descritos en la sesión de teoría.

Puesto que el problema de estimación de los parámetros de un modelo de regresión lineal es un problema de optimización convexa sin restricciones, compararemos los valores de los parámetros ajustados con los que obtendríamos aplicando un método iterativo de optimización como el método de descenso de gradiente visto en la sesión anterior. También comentaremos un método alternativo al método de descenso de gradiente, denominado método de descenso de gradiente estocástico (Stochastic Gradient Descent).

El modelo lineal presenta ventajas desde el punto de vista de la estimación y la interpretación, sin embargo, existen situaciones en las que los estimadores de los parámetros del modelo (obtenidos mediante el método de mínimos cuadrados) podrían no resultar adecuados, dando lugar por ejemplo a una baja precisión en las predicciones. Una de las situaciones en las que surge este problema es cuando los predictores están fuertemente correlacionados. Otra situación que afecta a la precisión de la predicción del modelo lineal y también a la falta de interpretabilidad del mismo, se da cuando el número de variables explicativas p es mayor que el número de observaciones n. La regresión Ridge y Lasso son dos modelos de regresión regularizada. Este tipo de métodos nos permiten mejorar la precisión de predicción del modelo lineal en situaciones como las citadas anteriormente. En esta práctica revisaremos como ajustar con R modelos de regresion lineal regularizada.

1 Ajuste de un modelo de regresión lineal con R

El fichero Advertising.csv contiene información sobre las ventas de un producto en 200 mercados diferentes, junto con la inversión en publicidad de dichos mercados en distintos medios. En primer lugar, importa los datos y realiza una representación gráfica que te permita visualizar la posible relación entre las ventas y la inversión en publicidad en distintos medios.



Vamos a asumir que la variable Y (`Sales`) depende linealmente de tres variables X_1 (`TV`), X_2 (`Radio`) y X_3 (`Newspaper`), es decir, planteamos el modelo

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon.$$

Para realizar un ajuste lineal en R, utilizaremos la función `lm`. El argumento principal de la función `lm` es una **fórmula** de R. En nuestro caso escribiremos:

```
> z <- lm(Advertising$Sales ~ Advertising$TV + Advertising$Radio + Advertising$Newspaper)
```

De forma equivalente, podríamos escribir:

```
> z <- lm(Sales ~ TV + Radio + Newspaper, data = Advertising)
```

Observa que no necesitamos especificar en la fórmula el término independiente, ya que éste se incluye por defecto. La función `lm` devuelve un objeto de tipo `lm` con varias componentes.

```
> class(z)
```

```
## [1] "lm"
```

```
> names(z)
```

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

Podemos resumir los resultados del ajuste con la funcion `summary`

```
> summary(z)
```

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

```
## lm(formula = Sales ~ TV + Radio + Newspaper, data = Advertising)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8277 -0.8908  0.2418  1.1893  2.8292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.938889   0.311908   9.422   <2e-16 ***
## TV           0.045765   0.001395  32.809   <2e-16 ***
## Radio        0.188530   0.008611  21.893   <2e-16 ***
## Newspaper   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.686 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

1.1 Estimación de los parámetros del modelo

Para obtener los coeficientes estimados del modelo, podemos utilizar la función `coef`

```
> coef(z)
```

```
##  (Intercept)           TV        Radio    Newspaper
##  2.938889369  0.045764645  0.188530017 -0.001037493
```

El resultado es equivalente a:

```
> z$coefficients
```

```
##  (Intercept)           TV        Radio    Newspaper
##  2.938889369  0.045764645  0.188530017 -0.001037493
```

Comprueba que los coeficientes ajustados por el método de mínimos cuadrados se obtienen como[1]:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{y}$$

donde, en nuestro ejemplo,

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & x_{13} \\ 1 & x_{21} & x_{22} & x_{23} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & x_{n3} \end{pmatrix}.$$

Para obtener los valores ajustados por el modelo de regresión lineal, $\hat{y}_i$, usaremos la función `fitted`

```
> fitted(z)
```

Podemos obtener la misma información escribiendo `z$fitted.values`. Los residuos del modelo, $\hat{\epsilon}_i$, serán por lo tanto

```
> Advertising$Sales - z$fitted.values
```

Equivalentemente,

[1] La función `solve` calcula la inversa de una matriz y la función `t` calcula la traspuesta

```
> residuals(z)
```

En el modelo de regresión lineal múltiple, el RSE (residual standard error) se calcula como:

$$RSE = \sqrt{\frac{RSS}{n-p-1}}.$$

Se trata de una estimación de la desviación típica del error. Calcula el RSE a partir de los residuos del modelo y comprueba que obtienes el mismo valor que se muestra al usar la función `summary`.

Podremos obtener intervalos de confianza para los coeficientes del modelo con la función `confint`. Si no se especifica el argumento `level`, los intervalos se calculan con una confianza $1 - \alpha = 0.95$.

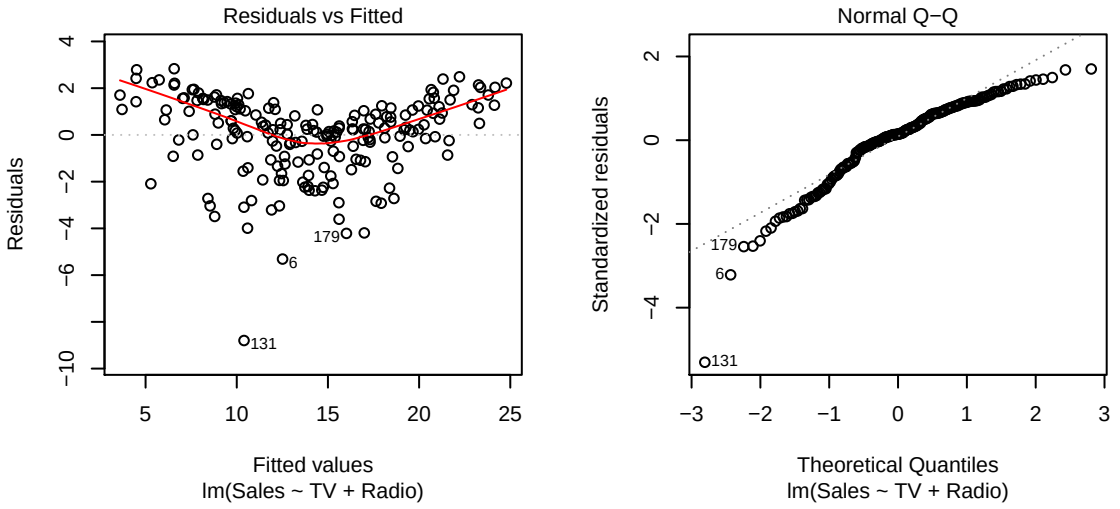
```
> confint(z, level = 0.9)
```

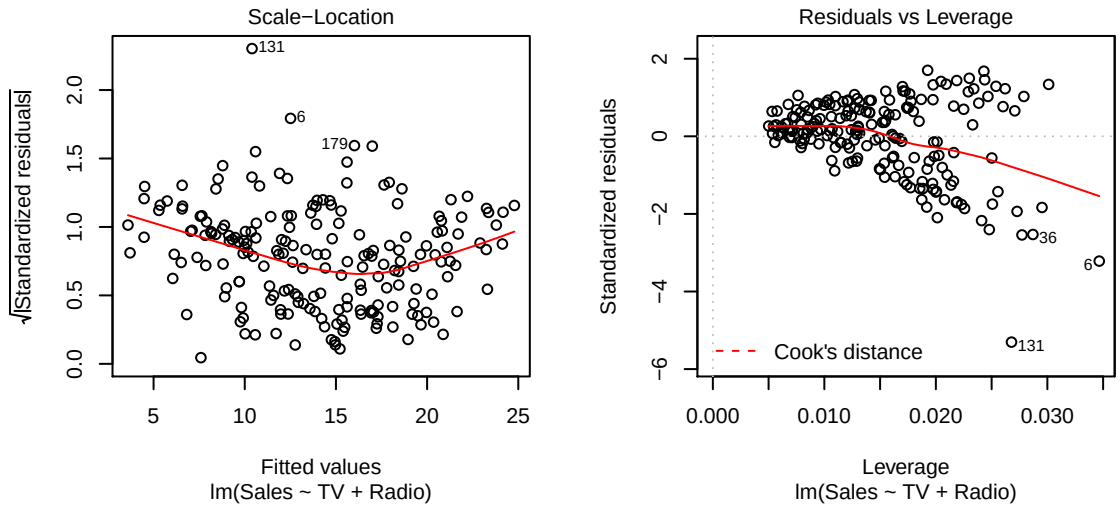
```
##                    5 %        95 %
## (Intercept)  2.42340953 3.454369213
## TV           0.04345935 0.048069943
## Radio        0.17429853 0.202761502
## Newspaper   -0.01074031 0.008665319
```

1.2 Contrastes sobre los parámetros del modelo

Analiza los resultados de los contrastes sobre los parámetros del modelo que devuelve la función `summary`. Observa que de los resultados se desprende (test F) que al menos una de las variables predictoras es útil para predecir la respuesta Y. Además, si nos fijamos en los contrastes individuales, parece que la variable `Newspaper` no es significativa. Ajusta un modelo lineal que explique la variable `Sales` como función de `TV` y `Radio` y analiza si se observa una pérdida importante en el ajuste del modelo (coeficiente R^2). Como verás, parece que el modelo que explica la variable `Sales` como función de `TV` y `Radio` es más razonable.

```
> z2 <- lm(Sales ~ TV + Radio, data = Advertising)
> plot(z2)
```





1.3 Predicción

Una vez fijado el modelo que explica la variable `Sales` como función de `TV` y `Radio`, podemos hacer predicciones como se muestra a continuación. Por ejemplo, si queremos predecir las ventas en un comercio que invierte 20000$ en publicidad en radio y 100000$ en publicidad en TV, escribiremos:

```
> newdata = data.frame(TV = 100, Radio = 20)
> predict(z2, newdata)
```

```
##        1
## 11.25647
```

A continuación, se muestra el intervalo de confianza para la venta media en mercados con una inversión de 20000$ en publicidad en radio y 100000$ en publicidad en TV. En segundo lugar se muestra el intervalo de predicción para la venta en un mercado que invierte 20000$ en publicidad en radio y 100000$ en publicidad en TV. Observa que el intervalo para la predicción es mayor.

```
> newdata = data.frame(TV = 100, Radio = 20)
> predict(z2, newdata, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 11.25647 10.98525 11.52768
```

```
> predict(z2, newdata, interval = "predict")
```

```
##        fit      lwr      upr
## 1 11.25647 7.929616 14.58332
```

2 Ajuste de los parámetros de un modelo de regresión lineal mediante métodos de optimización

Consideremos de nuevo el modelo de regresión lineal

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon.$$

Recordamos que, dada una muestra de entrenamiento $(y_1, x_{11}, \ldots, x_{1p}), \ldots, (y_n, x_{n1}, \ldots, x_{np})$, los parámetros estimados del modelo se obtienen minimizando la función:

$$RSS(\beta_0, \beta_1, \ldots, \beta_p) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip})^2. \tag{1}$$

El problema de estimación de los parámetros se puede ver, por lo tanto, como un problema de optimización convexa sin restricciones. Podremos aproximar su solución mediante un método de optimización como, por ejemplo, el método de descenso de gradiente.

2.1 Método de descenso de gradiente

Para simplificar el problema, consideraremos un modelo de regresión lineal simple:

$$Y = \beta_0 + \beta_1 X_1 + \epsilon.$$

En primer lugar, vamos a simular una muestra con $n = 100$ observaciones, (y_i, x_i), de un modelo de regresión lineal simple con parámetros β_0 y β_1. Para ello, generamos en primer lugar los valores x_i a partir de una distribución uniforme. A continuación generamos los errores del modelo, ϵ_i, a partir de una distribución normal de media 0 y varianza σ^2. Los valores y_i se calcularán entonces como

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i.$$

En R, haremos:

```
> n <- 100
> x <- runif(n, min = 0, max = 5)  # x_i: n puntos aleatorios en el itervalo [min,max]
> beta0 <- 2  # Parámetro beta0 del modelo
> beta1 <- 5  # Parámetro beta1 del modelo
> epsilon <- rnorm(n, sd = 1)  # error (con desviación típica sd=1)
> y <- beta0 + beta1 * x + epsilon  # y_i
```

Puedes hacer un diagrama de dispersión de la muestra de entrenamiento generada y ver el efecto de diferentes valores del parámetro `sd`. Calcula el valor de los parámetros estimados con la función `lm`.

Veremos ahora como aproximar los parámetros del modelo mediante el método de descenso de gradiente (también conocido como batch gradient descent). Para ello debemos minimizar la función:

$$J(\beta_0, \beta_1) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2.$$

Observa que este problema es equivalente a (1). El factor 1/2 es únicamente para simplificar la notación del método de descenso de gradiente.

Comprueba que el método de gradiente para minimizar la función $J(\beta_0, \beta_1)$ se puede escribir:

Algoritmo: Método de descenso de gradiente para $J(\beta_0, \beta_1)$

Dado $(\hat{\beta}_0, \hat{\beta}_1)$ y $t > 0$

repite

$\hat{\beta}_0 = \hat{\beta}_0 + t \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)$

$\hat{\beta}_1 = \hat{\beta}_1 + t \sum_{i=1}^{n} x_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)$

hasta que se cumpla el criterio de parada

Prográmalo en R y compara los resultados obtenidos con los que devuelve la función `lm`. Recuerda que el método de descenso de gradiente converge (siempre que el paso t no sea demasiado grande).

2.2 Método de descenso de gradiente estocástico (Stochastic Gradient Descent)

Observa que en el método de descenso de gradiente, para cada actualización del valor de los parámetros estimados, se necesita la muestra de entrenamiento al completo. Existe una alternativa al método de descenso de gradiente que también proporciona buenos resultados. El algoritmo para el caso particular del modelo de regresión lineal simple es el siguiente:

Algoritmo: Método de descenso de gradiente estocástico para $J(\beta_0, \beta_1)$

Dado $(\hat{\beta}_0, \hat{\beta}_1)$ y $t > 0$
repite
  repite para cada $i = 1, \ldots, n$
  $\hat{\beta}_0 = \hat{\beta}_0 + t(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)$
  $\hat{\beta}_1 = \hat{\beta}_1 + t x_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)$
hasta que se cumpla el criterio de parada

Observa que con este método se actualiza el valor de los parámetros con cada observación de la muestra de entrenamiento. Mientras que el método de descenso de gradiente (batch) necesita leer todos los datos para hacer una única iteración (lo cual puede ser costoso si n es muy grande), el método de descenso de gradiente estocástico puede empezar a iterar desde la primera observación.

3 Problemas en el ajuste del modelo de regresión lineal

Como acabamos de comentar, existen situaciones en las que los estimadores de los parámetros del modelo de regresión lineal (obtenidos mediante el método de mínimos cuadrados) podrían no resultar adecuados. Analizaremos en esta sección dos situaciones habituales que provocan inestabilidad en los parámetros estimados.

3.1 Efecto de la multicolinealidad en la estimación del modelo de regresión lineal por el método de mínimos cuadrados

En primer lugar analizaremos mediante un pequeño estudio de simulación, como la multicolinealidad afecta de forma importante a la varianza de los estimadores del modelo de regresión lineal.

Para ello, generaremos muestras de entrenamiento del modelo

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

siendo X_1, X_2 variables con distintos niveles de correlación.

Dadas dos variables aleatorias X_1 y X_2, podemos medir la relación lineal que hay entre ambas variables mediante el coeficiente de correlación, definido por

$$\rho = \frac{\mathrm{Cov}(X_1, X_2)}{\sigma_{X_1} \sigma_{X_2}},$$

donde $\mathrm{Cov}(X_1, X_2)$ denota la covarianza entre X_1 y X_2 y σ_{X_1}, σ_{X_2} denota la desviación típica de X_1 y X_2, respectivamente. El coeficiente de correlación entre dos variables satisface $-1 \leq \rho \leq 1$. Diremos que dos variables son incorreladas si $\rho = 0$.

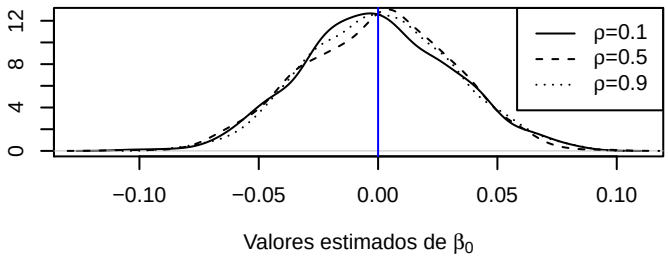
Se pueden simular fácilmente observaciones de dos variables con una determinada correlación ρ en R. Para ello, en primer lugar generamos dos secuencias de números aleatorios incorrelados, X_1 y X_1^* , a partir de una distribución normal (usando la función `rnorm`). A continuación se calcula $X_2 = \rho X_1 + \sqrt{1 - \rho^2} X_1^*$. Aplica el procedimiento descrito para generar con R observaciones de dos variables con distintas correlaciones ρ. Comprueba, con la función `cor`, que la correlación muestral de las secuencias X_1 y X_2 obtenidas se aproxima a la correlación ρ elegida.

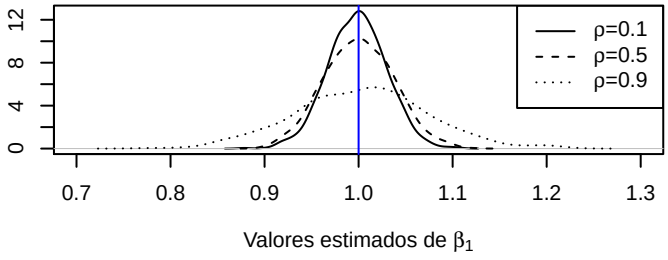
Para analizar el efecto de la correlación entre variables en la estimación de los parámetros de un modelo de regresión lineal, realiza el siguiente ejercicio.

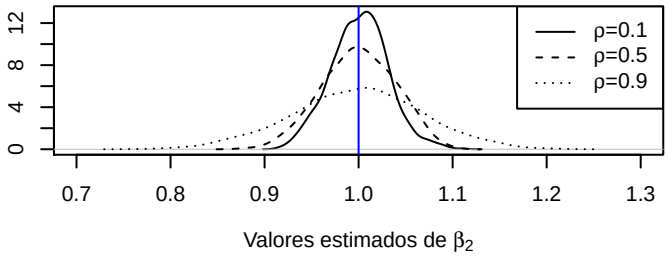
1. Genera una muestra de tamaño $n = 1000$ de dos variables X_1 y X_2 con correlación $\rho = 0.1$.
2. Genera $B = 1000$ muestras del modelo

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

   con $\beta_0 = 0$, $\beta_1 = 1$ y $\beta_2 = 1$.
3. Ajusta un modelo de regresión lineal a cada una de las muestras y guarda los parámetros estimados del modelo.
4. Repite el mismo procedimiento para niveles de correlación $\rho = 0.5$ y $\rho = 0.9$.
5. Para cada uno de los parámetros del modelo, representa en un mismo gráfico la distribución de los valores estimados para los distintos niveles de correlación elegidos. Puedes usar la función `density`.







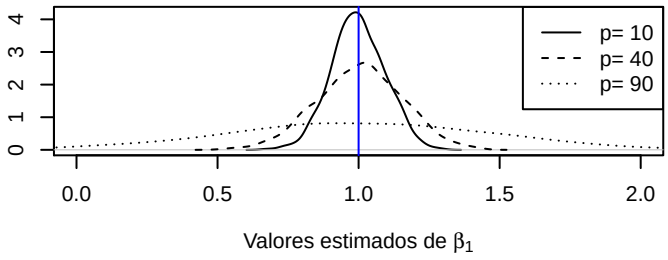
3.2 Efecto de la alta dimensión en la estimación del modelo de regresión lineal por el método de mínimos cuadrados

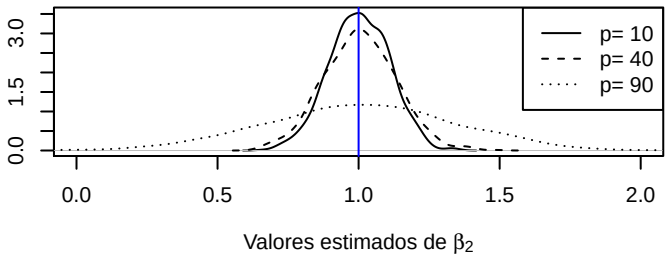
Otra de las situaciones en las que los estimadores de los coeficientes del modelo de regresión lineal son inestables se da cuando el número de variables explicativas p es elevado con respecto al número de observaciones n. Para analizar el efecto de la alta dimesión en la estimación de los parámetros de un modelo de regresión lineal, realiza el siguiente ejercicio.

1. Genera una muestra de tamaño $n = 100$ de $p = 90$ variables incorreladas.
2. Genera $B = 1000$ muestras del modelo

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon$$

   con $\beta_0 = 0$, $\beta_i = 1$, para $i = 1, \ldots, p$.
3. Ajusta un modelo de regresión lineal a cada una de las muestras y guarda las estimaciones de los parámetros $\hat{\beta}_1$ y $\hat{\beta}_2$.
4. Repite el mismo procedimiento para distintos valores de p, por ejemplo, $p = 40$ y $p = 10$.
5. Para $\hat{\beta}_1$ y $\hat{\beta}_2$, representa en un mismo gráfico la distribución de los valores estimados de para los distintos valores de p elegidos. Puedes usar la función `density`.





4 Regresión lineal con regularización

4.1 Regresión Ridge

Son varios los paquetes disponibles en R para el ajuste del modelo de regresión lineal mediante técnicas de regularización. Por ejemplo, para el caso de la regresión Ridge podremos usar, entre otros, los paquetes `MASS` y `glmnet`.

Trabajaremos con los datos discutidos en la sesión de teoría correspondientes al estudio sobre cáncer de próstata de Stamey et al. (1989). Los datos se encuentran disponibles en la librería `ElemStatLearn`.

```
> library(ElemStatLearn)  # Prostate data set
> data(prostate)
```

Los datos aparecen recogidos en un `data.frame` con 10 variables y 97 observaciones. El objetivo del estudio es determinar qué variables influyen en la presencia de un antígeno prostático específico (variable `lpsa`) para detectar el cáncer de próstata. Disponemos así como variables explicativas de medidas como el volumen del tumor (`lcavol`), el log-peso de la próstata (`lweight`), la edad (`age`), la cantidad de hiperplasia prostática benigna (`lbph`), el grado de infiltración del tumor en la vesícula seminal (`svi`), el grado de penetración capsular (`lcp`) y el score de Gleason (`gleason` y `pgg45`). Además disponemos de una variable (`train`) que divide la muestra en una muestra de entrenamiento ($n = 67$) y una muestra test ($n = 30$).

Proponemos un modelo de regresión lineal para explicar la variable `lpsa` en función del resto de marcadores clínicos ($p = 8$).

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon$$

En primer lugar, selecciona las observaciones correspondientes a la muestra de entrenamiento.

Ajusta un modelo de regresión lineal que explique la variable `lpsa` en función del resto de variables del modelo, utilizando el método de estimación de mínimos cuadrados (OLS). Recuerda que estimador por el método de mínimos cuadrados de los parámetros $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^t$ se obtienen resolviendo el problema de optimización:

$$\underset{\boldsymbol{\beta}}{\text{Minimizar}} \quad \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip})^2$$

A continuación se muestran los valores de los parámetros estimados.

	OLS
intercept	0.4292
xlcavol	0.5765
xlweight	0.6140
xage	-0.0190
xlbph	0.1448
xsvi	0.7372
xlcp	-0.2063
xgleason	-0.0295
xpgg45	0.0095

Ajustaremos ahora el mismo modelo de regresión lineal, mediante el procedimiento de estimación Ridge. Recuerda que en ese caso, los parámetros $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^t$ se obtienen resolviendo el problema de optimización

$$\underset{\boldsymbol{\beta}}{\text{Minimizar}} \quad \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip})^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

La solución del problema depende del valor de λ seleccionado. En particular, para $\lambda = 0$ el problema se reduce al estimador ordinario de mínimos cuadrados. Para realizar el ajuste mediante el procedimiento de estimación Ridge con R, usamos la función `lm.ridge` del paquete `MASS`. Observa que la sintaxis es similar a la de la función `lm`, añadiendo el argumento `lambda` con el valor de la penalización λ que queramos considerar.

```
> library(MASS)
> rr <- lm.ridge(y ~ x, lambda = 0)
```

En cuanto a los parámetros estimados, debemos distinguir entre el resultado almacenado en `rr$coef` y los valores que nos devuelve `coef(rr)`. Puedes observar que el resultado de `coef(rr)` coincide con el obtenido por el método de mínimos cuadrados. La diferencia con respecto a los valores almacenados en `rr$coef` es que éstos no están en la escala original. Son los coeficientes que se obtienen de ajustar el modelo de regresión a las variables predictoras estandarizadas. Es decir, coincidirían con los resultados de ajustar el modelo de regresión lineal por el método de mínimos cuadrados a los datos que se obtienen tras restar a cada columna de la matriz `x` su media y dividir entre la desviación típica. Para comprobarlo, puedes utilizar la función `scale`, que estandariza una matriz de datos por columnas.

A continuación realizamos el ajuste mediante el procedimiento de estimación Ridge, usando diferentes valores del parámetro λ. Recuerda que el incremento del valor de λ implica la contracción del vector $(\hat{\beta}_1, \ldots, \hat{\beta}_p)^t$.

```
> lam <- seq(0, 10, by = 0.01)
> rr <- lm.ridge(y ~ x, lambda = lam)
```

Puedes observar que ahora `coef(rr)` nos devuelve una matriz donde las filas recogen los parámetros estimados para cada valor de λ.

Otro de los paquetes de R que nos permiten realizar un ajuste lineal mediante el procedimiento de estimación Ridge es el paquete `glmnet`. El paquete `glmnet` es uno de los más utilizados en el contexto de regresión regularizada, ya que implementa en una única función (la función `glmnet`) distintos tipos de penalización. Veremos como es el uso de la función `glmnet` para regresión Ridge.

```
> library(glmnet)
> rr_glmnet <- glmnet(x, y, alpha = 0, lambda = 0)
```

El argumento `alpha` nos permite especificar el tipo de regularización que deseamos usar (`alpha=0` para Ridge). De nuevo, el argumento `lambda` corresponde al parámetro de penalización. Puedes observar con `coef(rr_glmnet)` que, como era de esperar, volvemos a obtener los mismos coeficientes estimados que con el ajuste por mínimos cuadrados. Usando diferentes valores del parámetro λ:

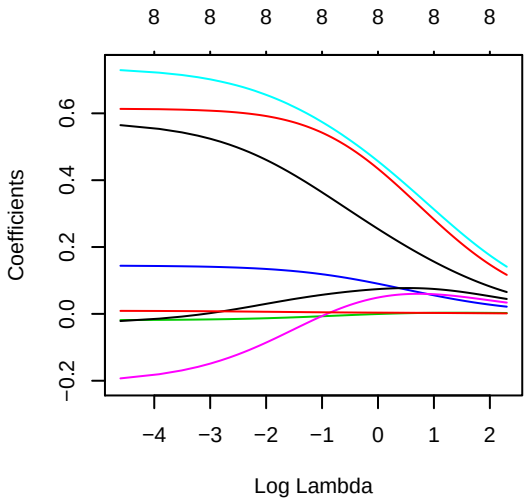
```
> lam <- seq(0, 10, by = 0.01)
> rr_glmnet <- glmnet(x, y, alpha = 0, lambda = lam)
```

Ahora, `coef(rr_glmnet)` nos devuelve una matriz donde las columnas recogen los parámetros estimados para cada valor de λ (las estimaciones se devuelven en la escala original). Fíjate que `coef(rr_glmnet)[,j]` son las estimaciones para el valor de λ almacenado en `rr_glmnet$lambda[j]`. Además, es importante mencionar que el problema de optimización que resuelve la función `glmnet` para Rdige es:

$$\underset{\boldsymbol{\beta}}{\text{Minimizar}} \quad \frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip})^2 + \frac{\lambda}{2}\sum_{j=1}^{p}\beta_j^2.$$

Observa que no coincide exactamente con la formulación original de problema y esto hace que un mismo valor de $\lambda \neq 0$ no devuelva los mismos estimadores con la función `lm.ridge` y `glmnet`. Aún así el papel de λ como parámetro de penalización es el mismo, como se observa en la siguiente gráfica, que muestra las estimaciones de los parámetros a medida que aumenta λ.

```
> plot(rr_glmnet, xvar = "lambda")
```



4.2 Regresión Lasso (least absolute shrinkage and selection operator)

Al igual que ocurre con la regularización Ridge, son varios los paquetes de R que implementan el método de reularización Lasso. Nosotros realizaremos el ajuste con la función `glmnet` que, como hemos dicho, implementa distintos tipos de penalización dependiendo del valor del argumento `alpha` que seleccionemos. Antes de realizar el ajuste, recordamos que en este caso los parámetros $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^t$ se obtienen resolviendo el problema de optimización

$$\underset{\boldsymbol{\beta}}{\text{Minimizar}} \quad \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip})^2 + \lambda \sum_{j=1}^{p} \mid \beta_j \mid .$$

Ajustamos el modelo con la función `glmnet`

```
> library(glmnet)
> lasso <- glmnet(x, y, alpha = 1, lambda = 0)
```

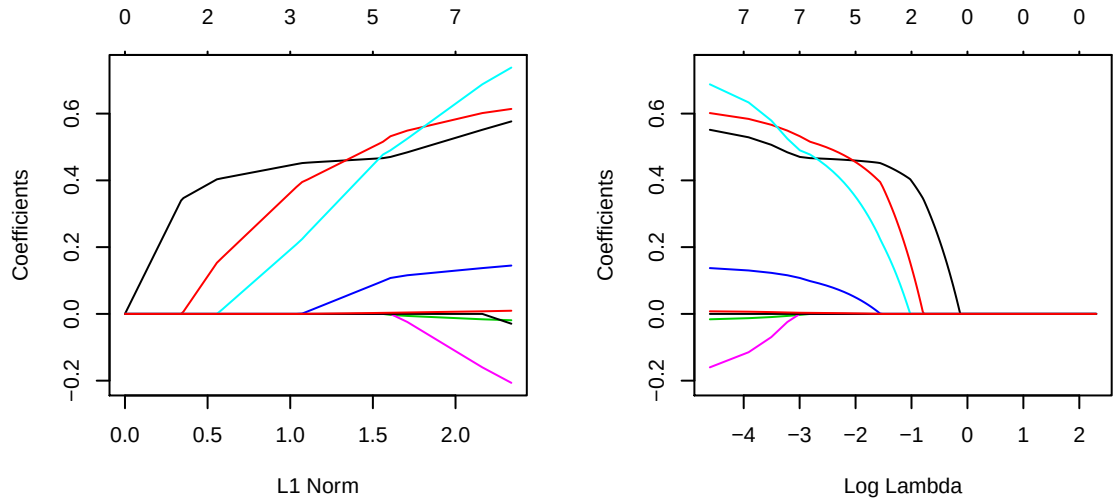
El argumento `alpha` nos permite especificar el tipo de regularización que deseamos usar (`alpha=1` para Lasso). Al igual que antes, el argumento `lambda` corresponde al parámetro de penalización y, de nuevo, `coef(lasso)` nos devuelve los coeficientes estimados que coinciden con los obtenidos por el ajuste por mínimos cuadrados. Usando diferentes valores del parámetro λ:

```
> lam <- seq(0, 10, by = 0.01)
> lasso <- glmnet(x, y, alpha = 1, lambda = lam)
```

Ahora, `coef(lasso)` nos devuelve una matriz donde las columnas recogen los parámetros estimados para cada valor de λ (las estimaciones se devuelven en la escala original)[2]. Representamos las estimaciones obtenidas en función de la norma L_1 del vector de parámetros estimado y en función del logaritmo de λ:

```
> plot(lasso)
> plot(lasso, xvar = "lambda")
```

[2]La función objetivo que minimiza `glmnet` para Lasso es $\frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip})^2 + \lambda \sum_{j=1}^{p} \mid \beta_j \mid .$



4.3 Selección del parámetro de penalización

Tanto el método de regularización Ridge como el Lasso, requieren un método para la selección del parámetro de penalización λ adecuado. Uno de los métodos más utilizados para este propósito es la *validación cruzada*. Aunque se hablará con más profundidad de este tipo de métodos en otros temas, comentaremos brevemente en qué consiste la validación cruzada en este contexto. El proceso fija un rango de posibles valores λ y ajusta el modelo para cada λ utilizando una parte de la muestra de entrenamiento (un subconjunto de observaciones se apartó previamente). A continuación se analiza la capacidad predictiva del modelo sobre el subconjunto de observaciones apartado (muestra de validación). Para ello es habitual calcular el error cuadrático medio (MSE) en la muestra de validación, es decir, $\frac{1}{n_v}\sum_{i=1}^{n_v}(y_i - \hat{y}_i)^2$, donde n_v es el tamaño de la muestra de validación, y_i son las observaciones de la muestra de validación e $\hat{y}_i$ son las predicciones dadas por el modelo ajustado. El λ óptimo es aquel para el cual el MSE sea menor.

Como este método depende en gran medida de qué observaciones se incluyen en la muestra para ajustar los modelos y cuáles en la muestra de validación, se han desarrollado otros métodos de validación cruzada.

Por ejemplo, la validación cruzada de k iteraciones (*k-fold cross-validation*) divide la muestra de entrenamiento en k subconjuntos de tamaño similar. En la primera etapa, se utiliza el primer subconjunto como muestra de validación, se ajusta el modelo en los $k-1$ subconjuntos restantes y se calcula MSE_1, el error cuadrático medio en la muestra de validación. A continuación se utiliza el segundo subconjunto como muestra de validación, se ajusta el modelo en los $k-1$ subconjuntos restantes y se calcula MSE_2, el error cuadrático medio en la muestra de validación. Por último, una vez repetido el mismo proceso para los k subconjuntos, se calcula la media aritmética de los MSE_i, con $i = 1, \ldots, k$. El error obtenido se denomina error de validación cruzada y el λ óptimo es aquel para el cual el error de validación cruzada sea menor. Un caso particular de validación cruzada de k iteraciones es $k = n$ (*Leave-one-out cross-validation*), donde las muestras de validación en cada iteración están formadas por una única observación.

La función `cv.glmnet` realiza la validación cruzada de k iteraciones para `glmnet` y nos devuelve el valor óptimo de λ según este criterio. Veamos como se ejecuta en primer lugar para la regularización Ridge.

```
> lam <- seq(0, 10, by = 0.01)
> cvout <- cv.glmnet(x, y, alpha = 0, lambda = lam)
```

La función `cv.glmnet` usa por defecto $k = 10$. El valor λ que minimiza el error de validación cruzada se encuentra almacenado en `cvout$lambda.min` y los parámetros obtenidos al ajustar el modelo de regresión

con regularización Ridge para ese valor de λ se obtienen directamente con la función `coef` como se muestra a continuación

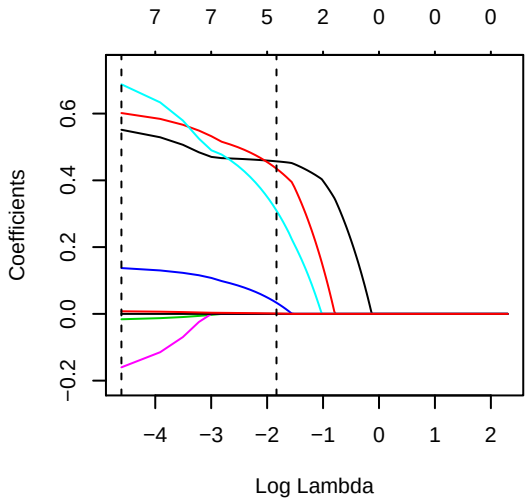
```
> cvout$lambda.min
> coef(cvout, s = "lambda.min")
```

La función también devuelve el valor `cvout$lambda.1se`, que representa un valor de λ para un modelo más simple que el de `cvout$lambda.min`, pero cuyo error está a una desviación estándar del mejor.

Del mismo modo, si queremos seleccionar el parámetro λ en el ajuste Lasso, haremos:

```
> lam <- seq(0, 10, by = 0.01)
> cvoutl <- cv.glmnet(x, y, alpha = 1, lambda = lam)
> cvoutl$lambda.min
> coef(cvoutl, s = "lambda.min")
> cvoutl$lambda.1se
> coef(cvoutl, s = "lambda.1se")
```

Mostramos a continuación una gráfica de los parámetros estimados con Lasso junto con el valor de λ seleccionado por validación cruzada (el valor óptimo `cvoutl$lambda.min` y `cvoutl$lambda.1se`)



En resumen, si seleccionamos como valores de λ los dados por `cvout$lambda.1se` (Ridge) y `cvoutl$lambda.1se` (Lasso), los parámetros estimados con mínimos cuadrados, Ridge y Lasso son los que se muestran en la siguiente tabla:

	OLS	Ridge	Lasso
intercept	0.4292	-0.1591	0.1769
xlcavol	0.5765	0.2692	0.457
xlweight	0.6140	0.4528	0.435
xage	-0.0190	-0.0012	.
xlbph	0.1448	0.0949	0.0334
xsvi	0.7372	0.4750	0.3094
xlcp	-0.2063	0.0440	.
xgleason	-0.0295	0.0725	.
xpgg45	0.0095	0.0038	0.001