# Regresión Lineal Múltiple-Parte 2: Diagnóstico, validación y Bootstrap

Departamento de Estadística e I.O.
Universitat de València

# Diagnóstico del modelo, validación y Bootstrap

# Gráficas de residuos

- Para linealidad y homocedasticidad, gráfica de residuos frente a valores ajustados
  - Mejor, residuos estandarizados: $e_i / \sqrt{\text{Var}(e_i)}$
    - $\text{Var}(e_i) = \hat{\sigma}^2(1 - h_i)$, $h_i$ elemento $ii$ de $H = X(X'X)^{-1}X'$
  - O mejor aún, residuos estudentizados: $e_i / \sqrt{\hat{\sigma_{-i}}^2(1 - h_i)}$
    - $\hat{\sigma_{-i}}^2$, es la $\hat{\sigma}^2$ obtenida eliminando el registro $i$ de la base y estimando el modelo con los $n - 1$ datos restantes.

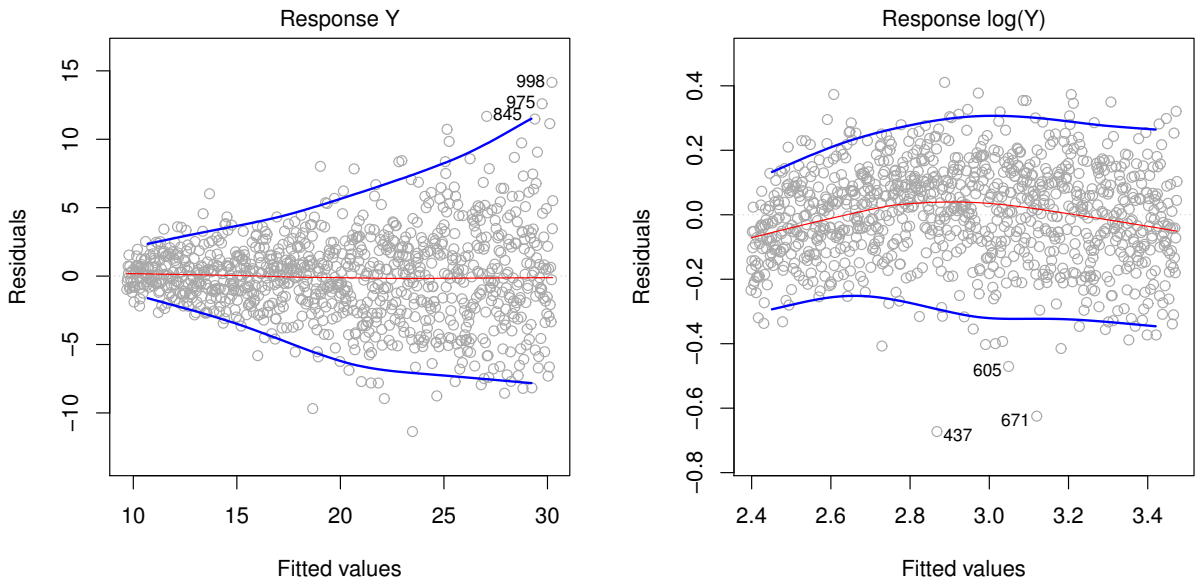
```
ml1 <- lm(sales ~ TV + radio, data= Advertising)
res_estandar <- rstandard(ml1)
res_student <- rstudent(ml1)
plot(ml1$fitted.values,res_student)
```

  - En valor absoluto, no deberían presentar patrones ni forma de embudo
- Las gráficas de residuos parciales son útiles para estudiar cada predictor por separado
  - Son los residuos al eliminar cada predictor

```
parciales <- residuals(ml1,type="partial")
plot(Advertising$TV,parciales[,1])
plot(Advertising$radio,parciales[,2])
```

# Heterocedasticidad

- Un ejemplo claro de heterocedasticidad




Fuente: Introduction to Statistical Learning, fig. 3.11

- Una transformación cóncava de $Y$ podría conseguir homocedasticidad, como $\sqrt{Y}$ o $\log(Y)$

# Transformación de Box Cox

- Si $Y$ es no negativa, las transformaciones Box-Cox son útiles en la búsqueda de homocedasticidad

$$g(y|\lambda) = \frac{y^\lambda - 1}{\lambda} \quad \text{si} \quad \lambda \neq 0, \qquad g(y|\lambda) = \lg(y) \quad \text{si} \quad \lambda = 0$$

  - Esta función es continua en $\lambda$
- El parámetro $\lambda$ se optimiza por máxima verosimilitud
  - Ejemplo: Datos `trees` en el paquete de R `MASS`

```
library(MASS)
ajuste_vol <- lm(Volume ~ Height,
                         data = trees)
boxcox(ajuste_vol, lambda = seq(-2,2, length = 10))
```

  - Produce un gráfico con la zona óptima de valores $\lambda$
- Transformar la variable respuesta $Y$ con el $\lambda$ óptimo

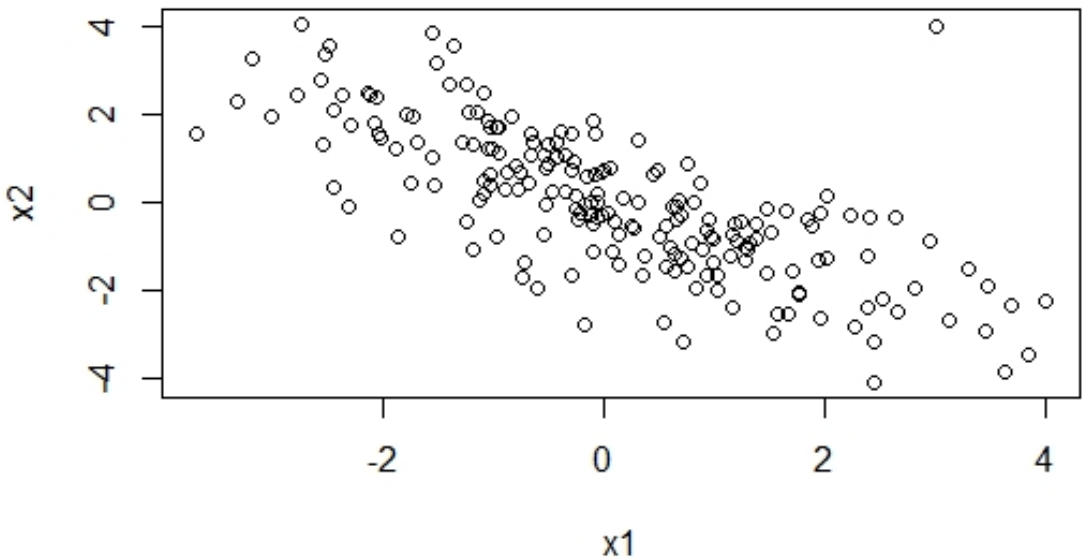
# Diagnóstico del modelo, validación y Bootstrap

# Outliers

- Una observación aberrante, un outlier, es una observación para la que su residuo es 'demasiado' grande
  - En la gráfica de residuos, son observaciones alejadas de la nube de puntos
  - Residuos estandarizados o estudentizados superiores a 3, en valor absoluto, son sospechosos
- Pueden no influir en el ajuste del modelo, pero siempre incrementan el error estándar de los residuos
  - Los intervalos de confianza y de predicción serán más amplios
- Pueden ser debidos a errores en los datos
  - Corregir errores si se comprueba que lo son
  - Dejadlos tal cual si no se descubre ningún error

# Observaciones influyentes

- Una observación es influyente si su exclusión tiene un impacto sustancial en el ajuste del modelo
- Son outliers respecto a los predictores
  - En regresión simple se detectan fácilmente: `boxplot()`
  - En regresión múltiple pueden ser difíciles de detectar en los gráficos



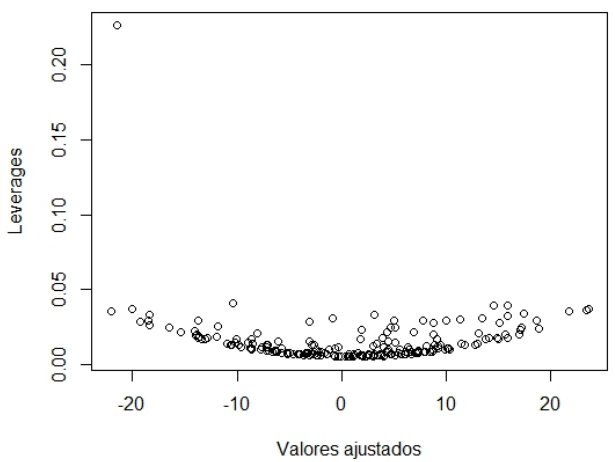
# Leverages

- El leverage de la observación $i$ es $h_i$, elemento $ii$ de $H$
  - Miden la importancia del cambio en $\hat{y}_i$ si cambia $y_i$, pues

$$h_i = \mathrm{d}\hat{y}_i(y)/\mathrm{d}y_i$$

  - Se utilizan para detectar observaciones influyentes
  - Ejemplo: Utilizando los datos del gráfico anterior

```
ajuste <- lm(y ~ x1 + x2)
plot(ajuste$fitted.values,hatvalues(ajuste))
```



# Distancia de Cook

- La distancia de Cook es otra medida de influencia

$$D_i = \frac{1}{(p+1)\hat{\sigma}^2} \sum_{j=1}^{n} (\hat{y}_j - \hat{y}_{j(i)})^2$$

$\hat{y}_{j(i)}$ es el valor ajustado al eliminar la observación $i$

  - Criterio teórico: La distancia de Cook es demasiado grande si $D_i > 1$. Puede ser muy laxo.
  - Ejemplo: Utilizando los datos del gráfico anterior

```
ajuste <- lm(y ~ x1 + x2)
plot(ajuste$fitted.values,cooks.distance(ajuste))
```

- El comando `plot(lm( ))` produce gráficas de residuos, normalidad y búsqueda de observaciones influyentes

# Diagnóstico del modelo, validación y Bootstrap

# Colinealidad

- La colinealidad es cuando dos o más predictores están muy correlacionados
  - Será difícil separar el efecto individual de cada predictor
  - Las varianzas de los predictores estarán infladas, afectando al test t y al intervalo de confianza
  - Ejemplo: Con los datos `Credit` del paquete `ISLR`, comparar los modelos

```
ajuste1 <- lm(Balance ~ Age + Limit , data=Credit)
ajuste2 <- lm(Balance ~ Age + Limit + Rating ,
                        data=Credit)
summary(ajuste1);summary(ajuste2)
```

- Correlación entre dos predictores se detecta fácilmente
  - Observar la matriz de correlaciones de los predictores
- Multicolinealidad, cuando hay involucrados más de dos predictores, es más difícil de detectar

# Factor de inflación de la varianza

- Factor de inflación de la varianza, VIF, para detectar multicolinealidad
  - $\text{VIF}_j = (1 - R_j^2)^{-1}$, donde $R_j^2$ coeficiente de determinación de la regresión de $X_j$, como variable respuesta, frente a los demás predictores
  - Se obtiene con el comando `vif()` del paquete `car`
    - Ejemplo: Con los datos Credit

```
library(car)
ajuste2 <- lm(Balance ~ Age + Limit + Rating ,
vif(ajuste2)
```

  - Debe preocuparnos que $\text{VIF}_j > 10$, o incluso $\text{VIF}_j > 5$
  - A la inversa de $\text{VIF}_j$ se le llama $\text{tolerancia}_j$
- Su justificación es porque, tras bastante algebra matricial:

$$\text{Var}(\hat{\beta}_j) = \sigma^2 (X'X)^{-1}_{jj} = \frac{\sigma^2}{(n-1)s_{x_j}^2} \frac{1}{1 - R_j^2} = \frac{\sigma^2}{(n-1)s_{x_j}^2} \text{VIF}_j$$

# Diagnóstico del modelo, validación y Bootstrap

# Error de ajuste versus error de predicción

- Error de ajuste Utilizamos el mismo banco de datos para estimar los parámetros del modelo y para medir su bondad
  - Puntos fuertes
    - Facilidad de uso. Las medidas de error de ajuste se pueden obtener siempre. Casi todas las que hemos propuesto hasta ahora son de este tipo
    - Soporte teórico. Son válidas si hay muchos datos y se dan las condiciones de aplicabilidad del modelo
  - Puntos débiles
    - Suelen subestimar la bondad de predicción del modelo
    - Puede darse un sobreajuste al seleccionar modelos
    - No están justificadas si no se cumplen las condiciones de aplicabilidad del modelo
- Error de predicción Si nuestro objetivo es la predicción, éste es el error que deberíamos medir, para poder minimizarlo
  - Se puede calcular si existe un segundo conjunto de datos: conjunto de validación

# Utilización de un conjunto de validación

- Si el banco de datos es grande, puede dividirse en dos:
  - Conjunto de entrenamiento Se utiliza para estimar los parámetros del modelo
  - Conjunto de validación Con él se calcula el error de predicción
- A este procedimiento se le denomina validación y al error cuadrático medio así obtenido error del conjunto de validación
  - Fortalezas Permite valorar la capacidad predictiva del modelo, sin preocuparnos por sobreajuste ni por las condiciones teóricas de aplicabilidad del modelo
  - Debilidades El error estándar del estimador 'error del conjunto de validación' es muy grande, a no ser que el tamaño muestral sea enorme, por lo que es poco fiable. Si se repite el proceso, los resultados pueden ser muy distintos

# Aproximación mediante conjunto de validación en R

- Un ejemplo: Datos `Credit` del paquete `ISLR`
  - Creamos los dos bancos de datos

```
datos <- subset(Credit,select = Income:Balance)
n <- nrow(datos)
seleccion <- sample(n,round(n/2))
entrenamiento <- datos[seleccion,]
prueba <- datos[-seleccion,]
```

  - Y ahora los analizamos

```
ajuste <- lm(Balance ~ . ,data=entrenamiento)
mean((prueba$Balance - predict(ajuste,prueba))**2)
```

- La fiabilidad del resultado es baja Repitiendo ese procedimiento, obtenemos resultados muy distintos:
  - Repitiendo 20 veces el ejemplo anterior

```
Min. 1st Qu. Median Mean 3rd Qu. Max.
8704   9587   10281 10212 10922 12157
```

# Diagnóstico del modelo, validación y Bootstrap

# Procedimientos de validación cruzada

- La aproximación mediante el conjunto de validación tiene dos inconvenientes
  - Utiliza un tamaño muestral bastante más pequeño que el del banco de datos original, para el ajuste del modelo y para la estimación del error de predicción
  - La estimación del error de predicción suele tener demasiada variabilidad
- Para disminuir esos efectos, los procedimientos de validación cruzada realizan varios análisis.
  - En cada análisis, el banco de entrenamiento es bastante mayor que el de prueba
  - Cada dato aparece una y solo una vez en un banco de prueba. Al final, hay un error de predicción para cada dato del banco original

# Procedimiento Leaving-one-out

- El procedimiento Leaving-one-out es el extremo opuesto al conjunto de validación
- Consiste en lo siguiente: Para cada dato $i = 1, \ldots, n$,
  - Ajusta el modelo eliminando el dato $i$, y obtiene su predicción $\tilde{y}_i$
  - Calcula su error cuadrático de predicción, $\text{MSE}_i = (y_i - \tilde{y}_i)^2$
  - Obtiene el error cuadrático medio: $\sum_i \text{MSE}_i / n$
- En general, es computacionalmente muy exigente. Hay que ajustar tantos modelos como datos
  - En modelos lineales es más sencillo, pues:

$$\tilde{e}_i = y_i - \tilde{y}_i = (y_i - \hat{y}_i)/(1 - h_i) = e_i/(1 - h_i)$$

Siendo $\tilde{y}_i$ el valor predicho sin utilizar el dato $i$ en el ajuste, $\hat{y}_i$ el valor predicho utilizando todos los datos, y $h_i$ el leverage de la observación $i$

# Leaving-one-out en R

- Este método se obtiene con el comando `cv.glm()` del paquete `boot`
  - Un ejemplo, con los datos `Credit`

```
library(boot)
datos <- subset(Credit,select = Income:Balance)
ajuste <- glm(Balance ~ ., data=datos)
cv.glm(datos,ajuste)$delta
```

  - Ese comando está preparado para modelos lineales generalizados, por lo que ajusta tantos modelos como datos. Es computacionalmente exigente
  - En este caso, al ser un modelo lineal, el error leaving-one-out se podría calcular de forma más eficiente:

```
ajuste <- lm(Balance ~ ., data=datos)
res <- ajuste$residuals; lev <- hatvalues(ajuste)
mean((res/(1-lev))**2)
```

# Validación cruzada en $k$ bloques

- $k$-Fold Cross-Validation es una opción intermedia entre el conjunto de validación y leaving-one-out
  - Consiste en dividir el banco de datos en $k$ bloques del mismo tamaño (aprox.) Realizar $k$ análisis, dejando un bloque fuera, como conjunto de validación, en cada análisis
  - Un ejemplo, siguiendo con los datos `Credit`:

```
library(boot)
datos <- subset(Credit,select = Income:Balance)
ajuste <- glm(Balance ~ ., data=datos)
cv.glm(datos, ajuste, K=8)$delta
```

- Características de los métodos de validación cruzada
  - Pueden utilizarse con cualquier tipo de modelos
  - Permiten estimar el error de predicción, que suele ser mayor, a veces bastante mayor, que el error de ajuste
  - Computacionalmente exigentes, pero asequibles con los ordenadores actuales

# Validación en selección de modelos

- Los métodos de validación y validación cruzada pueden utilizarse en selección de modelos, como alternativa a los criterios AIC, BIC, Cp de Mallow y $R^2$ ajustado
- Para ello, suele utilizarse un procedimiento mixto
  - El mejor modelo, para cada número de predictores dado, se elige utilizando errores de ajuste: suma de cuadrados de los residuos
    - Para ello, se puede utilizar `regsubsets()`, comando utilizado en Best Subset Selection
  - La comparación entre modelos de distinto tamaño se hace con errores de predicción, mediante validación o validación cruzada
    - Analizando solamente los modelos obtenidos con `regsubsets()`

# Diagnóstico del modelo, validación y Bootstrap

# Métodos Bootstrap

- Los métodos de remuestreo o bootstrap reutilizan los datos para construir múltiples bancos de datos que son analizados posteriormente
  - Validación cruzada es un caso particular de remuestreo
- Muy útiles para cuantificar la incertidumbre sobre un estimador o un método de aprendizaje estadístico
  - Obtención de errores estándar e intervalos de confianza
- Buena alternativa no paramétrica, fácil de implementar
- Su justificación teórica se basa en las propiedades en el muestreo de la distribución empírica de los datos
  - Por ello es conveniente que el tamaño muestral sea grande

# Errores estándar bootstrap

Para obtener el error estándar bootstrap de un estimador:

- Sea $\hat{\theta}$ la estimación de $\theta$ obtenida con la muestra original
- Para $b = 1, \ldots, B$, siendo $B$ grande, repetir:
  - A partir de la muestra original, se selecciona una muestra con reemplazamiento del mismo tamaño que la original
    - Notad que algunos datos pueden estar repetidos, otros no aparecer
  - Sea $\hat{\theta}_b$ la estimación de $\theta$ obtenida con esa muestra
- El sesgo se define como : $\hat{\theta} - \sum_{b=1}^{B} \hat{\theta}_b / B$
- El error estándar bootstrap de $\hat{\theta}$ se calcula:

$$\sqrt{\frac{1}{B-1} \sum_{b=1}^{B} (\hat{\theta} - \hat{\theta}_b)^2}$$

# Bootstrap: Un ejemplo

- Si los datos observados son bastante asimétricos, la mediana puede ser mejor medida de localización que la media
  - No es fácil obtener el error estándar paramétrico de la mediana, aunque se conozca la distribución de la que proceden los datos
  - Su error estándar bootstrap es fácil de calcular
- Ejemplo: Mediana de `Balance`, banco de datos `Credit`

```
mediana.datos <- median(Credit$Balance)
medianas.boot <- NULL
B <- 1000
for (i in 1:B) {
    indice <- sample(nrow(Credit),nrow(Credit),rep=TRUE)
    medianas.boot <- c(medianas.boot,
                median(Credit$Balance[indice]))}
sqrt(sum((mediana.datos-medianas.boot)**2)/(B-1))
```

# Bootstrap: Función `boot()`

- El paquete `boot` automatiza el análisis bootstrap
  - En el ejemplo anterior:

```
library(ISLR)
library(boot)
B <- 1000
boot.fun <- function(variable,indice)
                median(variable[indice])
boot(Credit$Balance,boot.fun,B)
```

- El comando `boot()` necesita:
  - Los datos originales `Credit$Balance`
  - La función que calcula el estadístico `boot.fun`
  - Y el número de repeticiones bootstrap `B`
  - Además, podrían utilizarse otros subcomandos
- Los métodos bootstrap utilizan simulación. Fijad una semilla aleatoria para reproducir los resultados

# Bootstrap: Otro ejemplo

- Error estándar de la recta de regresión en un punto. Datos `salarios.txt`, punto `experiencia = 5`
  - Leemos los datos y definimos alguna constante

```
salarios <- read.table("salarios.txt",header=T)
library(boot)
B <- 1000; x <- 5
```

  - Definimos la función que calcula la recta de regresión

```
boot.fun <- function(datos,indice,x) {
        coeficientes <- coef(lm(salario ~ experiencia,
                                data=datos, subset=indice))
        return(coeficientes[1]+coeficientes[2]*x)}
```

  - Ejecutamos la función `boot()`

```
boot(salarios,boot.fun,B,x=x)
```

- Si se sospecha que no se cumplen las condiciones de aplicabilidad del modelo lineal, la solución bootstrap es más adecuada