# SESSION 5 SUMMARY

Victor Miguel Terrón Macias

24/1/2021

## SESION 5. REGRESION LINEAL Y CLASIFICACIÓN

### INTRODUCCIÓN

Supongamos que nuestro trabajo consiste en aconsejar a un cliente sobre cómo mejorar las ventas de un producto particular, y el conjunto de datos con el que disponemos son datos de Publicidad que consisten en las ventas de aquel producto en 200 diferentes mercados, junto con presupuestos de publicidad para el producto en cada uno de aquellos mercados para tres medios de comunicación diferentes: TV, radio, y periódico. No es posible para nuestro cliente incrementar directamente las ventas del producto. Por otro lado, ellos pueden controlar el gasto en publicidad para cada uno de los tres medios de comunicación. Por lo tanto, si determinamos que hay una asociación entre publicidad y ventas, entonces podemos instruir a nuestro cliente para que ajuste los presupuestos de publicidad, y así indirectamente incrementar las ventas.

En otras palabras, nuestro objetivo es desarrollar un modelo preciso que pueda ser usado para predecir las ventas sobre la base de los tres presupuestos de medios de comunicación. En este contexto, los presupuestos de publicidad son las variables de entrada mientras que las ventas es una variable de salida. Las variables de entrada típicamente se denotan usando el símbolo X, con un subíndice para distinguirlas. Así X1 puede ser el presupuesto para TV, X2 el presupuesto para radio, y X3 el presupuesto para periódico. Las entradas tienen diferentes nombres, tales como predictores, variables independientes, características, o a veces solo variables. La variable de salida -en este caso, las ventas- frecuentemente es llamada la variable de respuesta o dependiente, y se denota típicamente con el símbolo Y.

Más generalmente, suponga que observamos una respuesta cuantitativa Y y p diferentes predictores, X1, X2, . . . , Xp. Asumimos que hay alguna relación entre Y y X=(X1, X2, . . . , Xp), la cual podemos escribir en la forma muy general

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

Figure 1: FORMULA

Aquí f es alguna función desconocida pero fija de X1, X2, . . . , Xp, y es un término de error aleatorio, el cual es independiente de X y tiene media cero. En esta formulación, f representa la información sistemática que X proporciona acerca de Y. Sin embargo, la función f que conecta las variables de entrada a la variable de salida en general es desconocida. En esta situación debemos estimar f basados en los datos observados. En esencia, el aprendizaje estadístico se refiere a un conjunto de enfoques para estimar f.

# ¿POR QUÉ ESTIMAR $f$?

Hay dos razones principales por las cuales podemos desear estimar f: predicción e inferencia.

## REGRESION LINEAL SIMPLE

Con frecuencia es necesario determinar si dos variables (aleatorias) están relacionadas de alguna manera. Por ejemplo, ¿tendrán los años de educación efecto sobre el salario que percibe un individuo? La relación entre dos variables cuantitativas puede visualizarse en un diagrama de dispersión en el plano, representando los valores de las variables en los ejes horizontal y vertical.

La correlación puede darse entre variables sin ninguna implicación de causalidad entre ellas, por ejemplo: si tomamos una muestra de individuos y medimos los diámetros del antebrazo y del muslo, seguramente encontraremos que hay una correlación positiva alta. Evidentemente no hay ninguna relación de causalidad entre estas variables y más bien ambas dependen del peso y la altura del individuo. A este tipo de correlación entre variables se le conoce como correlación espuria. La asociación más simple entre variables es cuando éstas se relacionan en forma lineal, sin embargo, no siempre es posible establecer este tipo de relación entre ellas. Para medir la magnitud de la asociación lineal entre dos variables, se utiliza comúnmente el coeficiente de correlación introducido por Karl Pearson. Éste es un número entre el -1 y el 1 denotado por la letra R. Si R = -1, se tiene una relación negativa perfecta y los puntos en el diagrama de dispersión se encuentran sobre una recta con pendiente negativa. Si R = 1, la relación lineal es también perfecta pero positiva: los puntos en el diagrama de dispersión están sobre una recta con pendiente positiva. Si R = 0, entonces no hay relación lineal alguna y los puntos forman más bien una nube difusa o algún otro patrón evidentemente no lineal. Lo usual es tener casos intermedios, en donde existe algún grado moderado de correlación lineal entre las variables. En general, en las ciencias sociales es raro tener coeficientes de correlación mayores que 0.7 (o menores que -0.7). A continuación, tenemos datos de estatura y pesos de unos individuos. Altura <- c(1.94, 1.82, 1.75, 1.80, 1.62, 1.64, 1.68, 1.46, 1.50, 1.55, 1.72, 1.67, 1.57, 1.60) Peso <- c(98, 80, 72, 83, 65, 70, 67, 47, 45, 50, 70, 61, 50, 52) Para obtener el coeficiente correlación de Pearson únicamente ejecutamos la siguiente instrucción en R cor(Altura, Peso) lo cual nos da 0.9645. A continuación, vamos a ajustar un modelo de regresión lineal simple a un conjunto de datos en R. Suponga que el conjunto de datos proviene de una fábrica que elabora productos

Para cada caso se considera un tamaño del proceso o tamaño de la ejecución (RunSize) y un tiempo del proceso o tiempo de la ejecución (RunTime). El tamaño del proceso representa la cantidad de artículos que se fabrica en un caso determinado, el tiempo del proceso representa la cantidad de minutos que toma elaborar los artículos en el caso especificado. En los datos anteriores, el primer caso indica que para elaborar 175 artículos se requirió un tiempo de 195 minutos. El segundo caso indica que para elaborar 189 artículos se tomó un tiempo de 215 minutos. El último caso indica que, para elaborar 68 artículos, se requirió un tiempo de fabricación de 172 minutos. Para comenzar a trabajar con los datos deberá guardarlos en su directorio de trabajo. A continuación, importe los datos a R mediante la siguiente instrucción production <- read.table("production.txt", header = TRUE), puede observar el conjunto de datos en R al ejecutar la palabra production. Extraiga las columnas RunSize y RunTime del data frame production mediante la instrucción attach(production), es decir, de ahora en adelante podrá utilizar los vectores RunSize y RunTime en R. Realice el gráfico de dispersión de los datos al ejecutar la siguiente instrucción plot(RunSize, RunTime, xlab="Run Size", ylab = "Run Time").

Cada punto del gráfico de dispersión representa el tamaño del proceso y el tiempo del proceso de un caso determinado. Ajuste un modelo de regresión lineal simple a los datos en R y obtenga un resumen del modelo

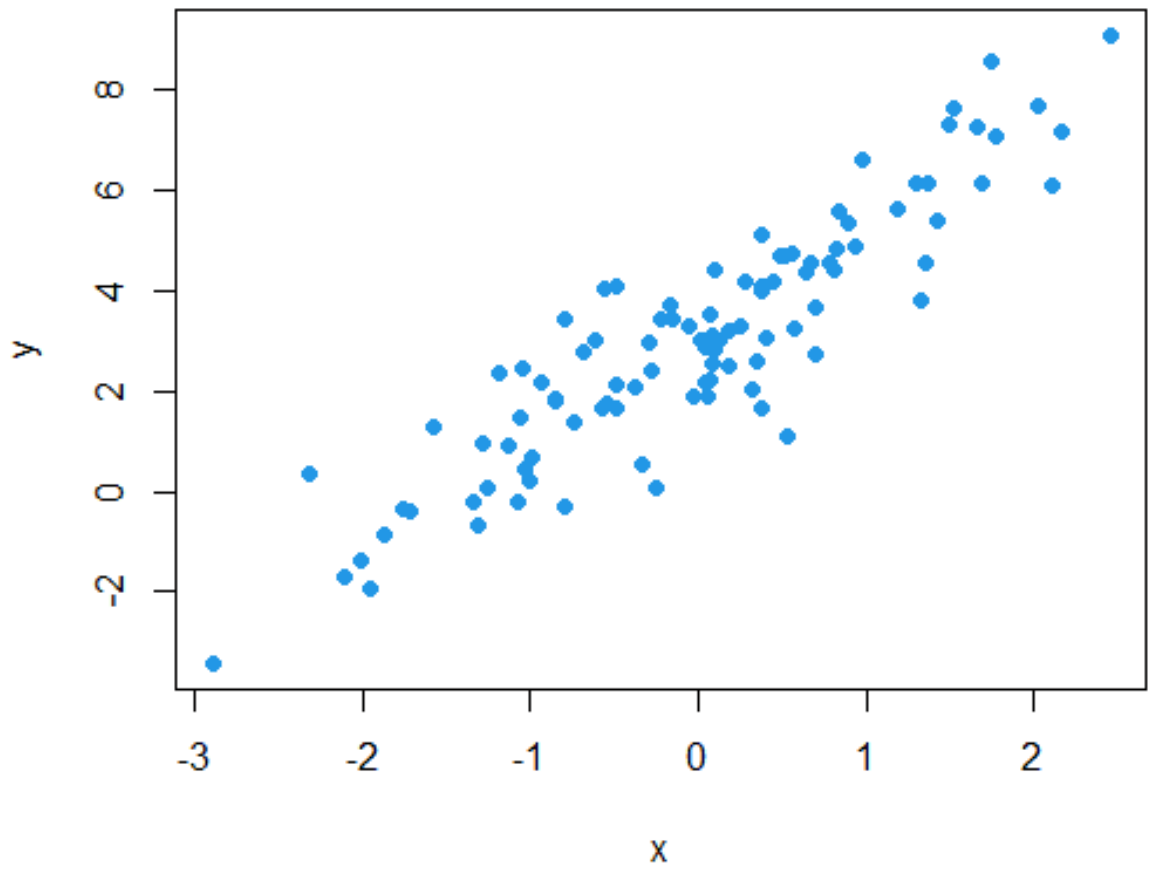



Figure 2: Ejemplo de regresion lineal

| Case | RunTime | RunSize |
|---|---|---|
| 1 | 195 | 175 |
| 2 | 215 | 189 |
| 3 | 243 | 344 |
| 4 | 162 | 88 |
| 5 | 185 | 114 |
| 6 | 231 | 338 |
| 7 | 234 | 271 |
| 8 | 166 | 173 |
| 9 | 253 | 284 |
| 10 | 196 | 277 |
| 11 | 220 | 337 |
| 12 | 168 | 58 |
| 13 | 207 | 146 |
| 14 | 225 | 277 |
| 15 | 169 | 123 |
| 16 | 215 | 227 |
| 17 | 147 | 63 |
| 18 | 230 | 337 |
| 19 | 208 | 146 |
| 20 | 172 | 68 |

Figure 3: tabla

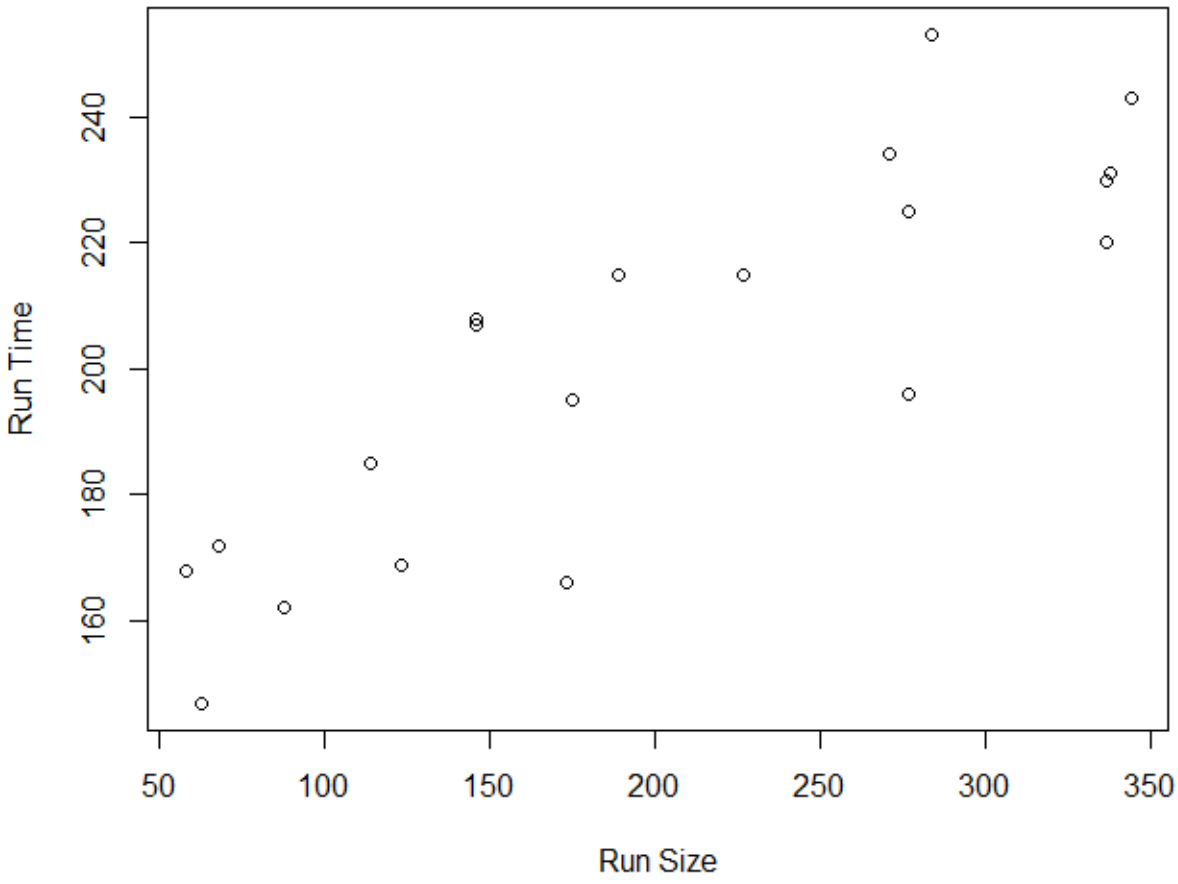



Figure 4: grafica

ajustado al ejecutar las siguientes dos instrucciones # Ajuste el modelo m1 <- lm(RunTime~RunSize) summary(m1)

```
Call:
lm(formula = RunTime ~ RunSize)

Residuals:
    Min      1Q  Median      3Q     Max
-28.597 -11.079   3.329   8.302  29.627

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 149.74770    8.32815   17.98 6.00e-13 ***
RunSize       0.25924    0.03714    6.98 1.61e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.25 on 18 degrees of freedom
Multiple R-squared:  0.7302,    Adjusted R-squared:  0.7152
F-statistic: 48.72 on 1 and 18 DF,  p-value: 1.615e-06
```

Figure 5: Imagen1

# MAQUINAS DE VECTORES DE SOPORTE

Un enfoque para clasificación que se desarrolló en la comunidad de las ciencias computacionales en los años 90 y que ha crecido en popularidad desde entonces son las máquinas de vectores de soporte (MVS o SVM por sus siglas en inglés). Las MVS han mostrado un buen desempeño en una variedad de contextos, y frecuentemente se les considera como uno de los mejores clasificadores.

## CLASIFICADOR DE MARGEN MAXIMO

Nuestro objetivo es desarrollar un clasificador basado en los datos de entrenamiento que clasificará una observación de prueba usando sus medidas características.

En un sentido, el hiperplano de margen máximo representa la línea media del bloque más ancho que podemos insertar entre las dos clases. Podemos calcular la distancia de cada observación de entrenamiento a un hiperplano de separación dado; la más pequeña de tales distancias es la distancia mínima de las observaciones al hiperplano y se conoce como el margen. El hiperplano de margen máximo es el hiperplano de separación para el cual el margen es el más grande-es decir, es el hiperplano que tiene la distancia mínima más lejana a las observaciones de entrenamiento-. En un espacio p-dimensional, un hiperplano es un subespacio plano de dimensión p-1 que no necesita pasar por el origen. En p dimensiones, un hiperplano se define por la ecuación

Podemos pensar al hiperplano como que divide el espacio p-dimensional en dos mitades. Por ejemplo, en dos dimensiones tenemos el hiperplano

El hiperplano de margen máximo es la solución al problema de optimización

sujeto a

La salida de R muestra entre muchas otras cosas el **intercepto estimado** $\hat{\beta}_0 = 149.7477$ y la **pendiente estimada** $\hat{\beta}_1 = 0.25924$ de la recta de regresión. Las fórmulas para obtener esos valores estimados a partir de los datos son las siguientes

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Donde los datos están dados por pares $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$.

Para graficar la recta de regresión estimada $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$, ejecute la siguiente instrucción en R

```
# Graficar la recta estimada

abline(lsfit(RunSize, RunTime))
```

Figure 6: Imagen2

## EL CASO NO SEPARABLE

Podemos extender el concepto de un hiperplano de separación para desarrollar un hiperplano que casi separa las clases usando lo que se conoce como un margen suave.

## CLASIFICADOR DE VECTORES DE SOPORTE

La distancia de una observación al hiperplano puede considerarse como una medida de nuestra confianza de que la observación se clasifica correctamente. Podemos estar dispuestos a considerar un clasificador basado en un hiperplano que no separe perfectamente las dos clases, con el interés de: * Mayor robustez a observaciones individuales, y Mejor clasificación de la mayoría de las observaciones de prueba.

Es decir, podría valer la pena clasificar mal unas pocas observaciones de entrenamiento para hacer un mejor trabajo al clasificar las observaciones restantes. Un hiperplano que casi separa las clases es la solución al problema de optimización.

Sujeto a:

M es el ancho del margen; buscamos hacer esta cantidad tan grande como sea posible. Una vez que hemos resuelto el problema de optimización, clasificamos una observación de prueba x* como antes, al simplemente determinar de que lado del hiperplano se encuentra. Es decir, clasificamos la observación de prueba basados en el signo de

Conforme el presupuesto C se incrementa, nos volvemos más tolerantes con respecto a las violaciones al margen, y así el margen se hará ancho. Por otro lado, cuando C decrece, nos volvemos menos tolerantes a las violaciones al margen y así el margen se hace angosto. En la práctica, C es tratada como un parámetro que generalmente se elige por medio de validación-cruzada. **Las observaciones que se encuentran directamente sobre los márgenes o del lado incorrecto del margen considerando su clase, se conocen como vectores de soporte. Clasificación con frontera de decisión no lineal.**

En el caso del clasificador de vectores de soporte, podemos tratar el problema de posibles fronteras no-lineales entre clases al ampliar el espacio de características usando funciones polinomiales cuadráticas, cúbicas, o

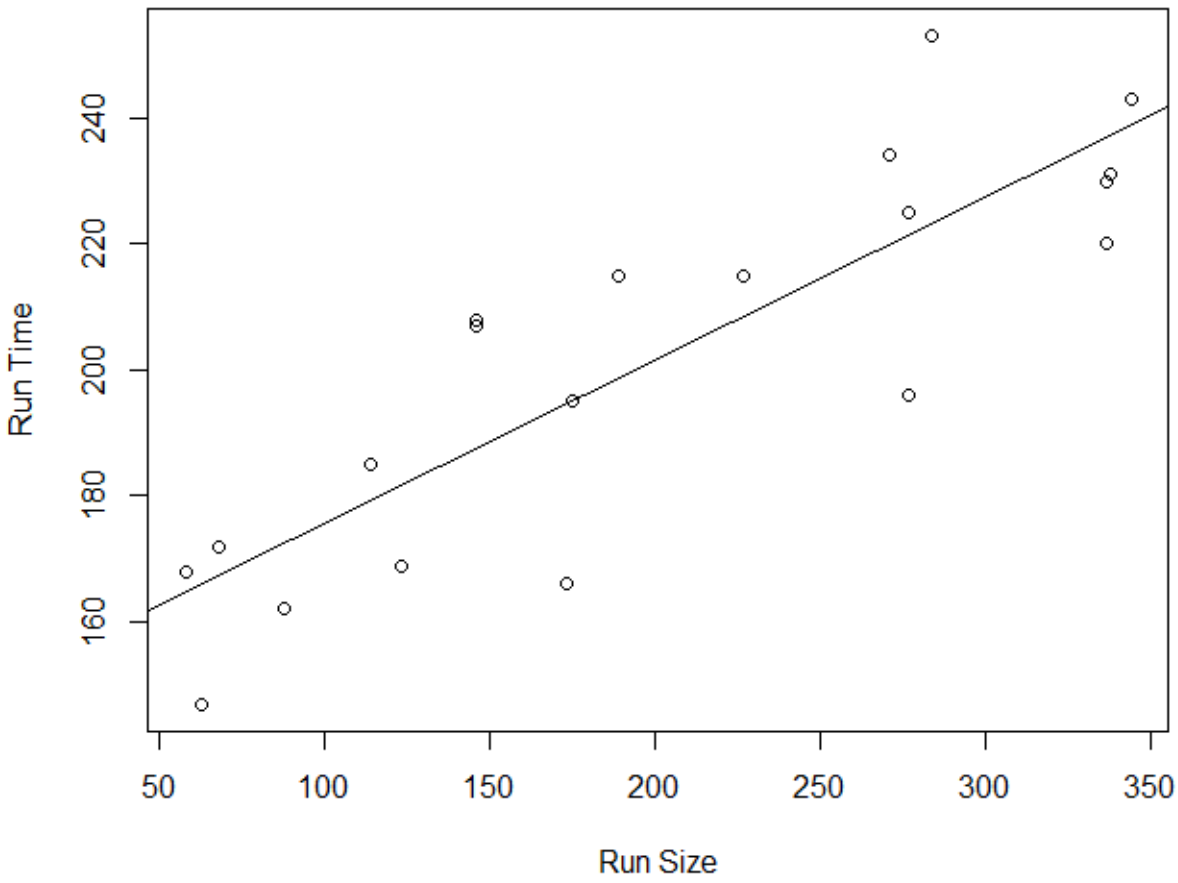



Figure 7: Imagen3

Los residuales del ajuste están definidos por $\hat{e}_i = y_i - \hat{y}_i$, es decir, a cada valor $y_i$ que tenemos, le restamos el valor estimado $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$. Para obtener los residuales del ajuste en R ejecute la siguiente instrucción m1$residuals

Figure 8: Imagen4

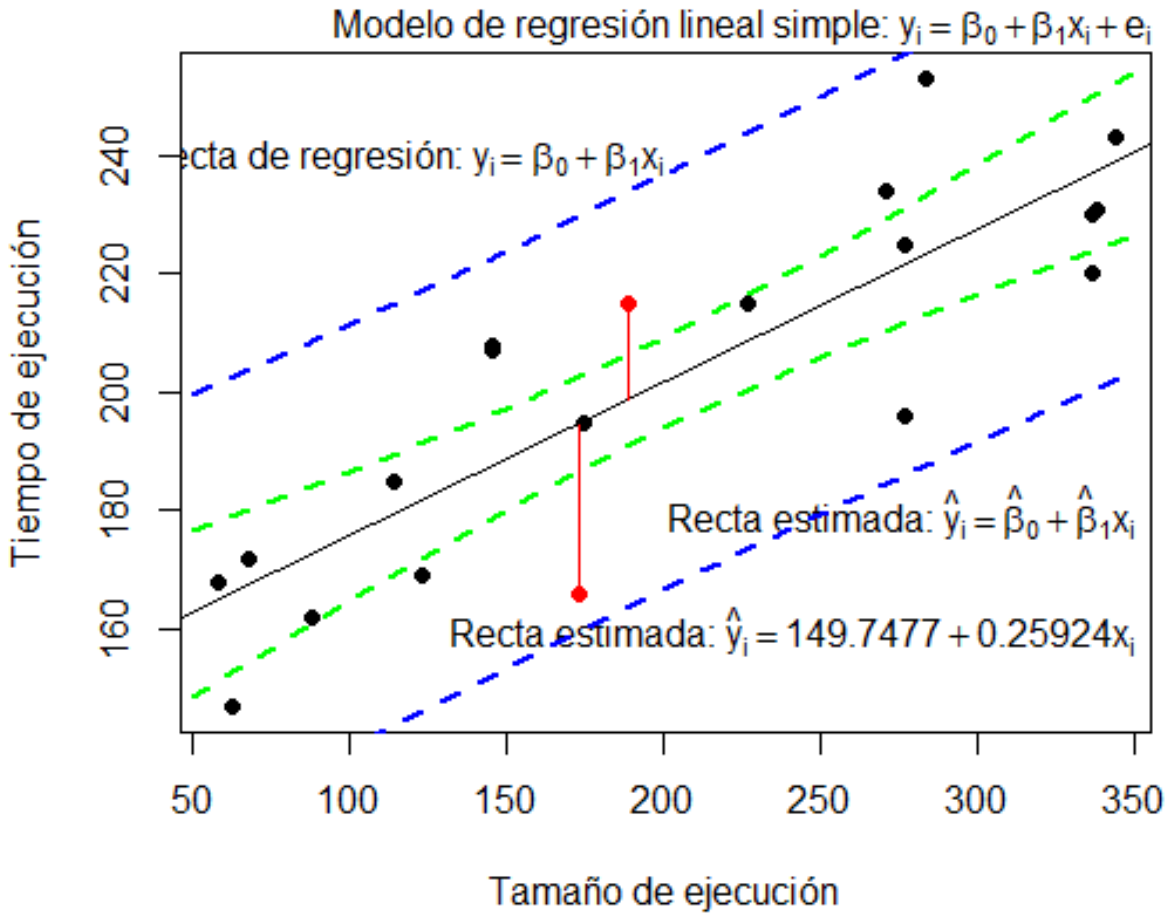



Figure 9: Imagen5

Obtenga la media de los residuales mediante mean(m1$residuals)

Obtenga una estimación de la varianza de los errores mediante sum(m1$residuals^2)/18

Figure 10: Imagen6

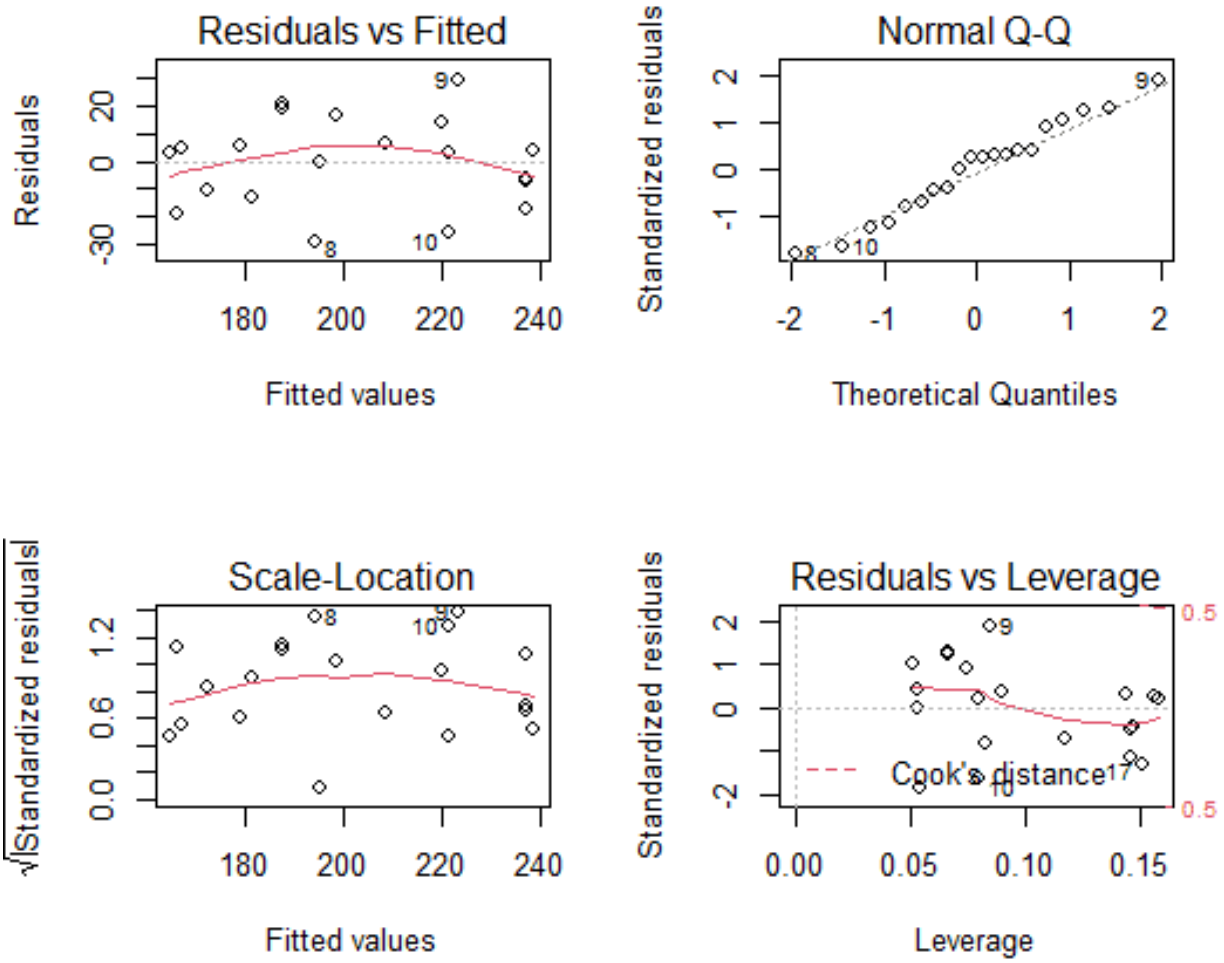



Figure 11: Imagen7

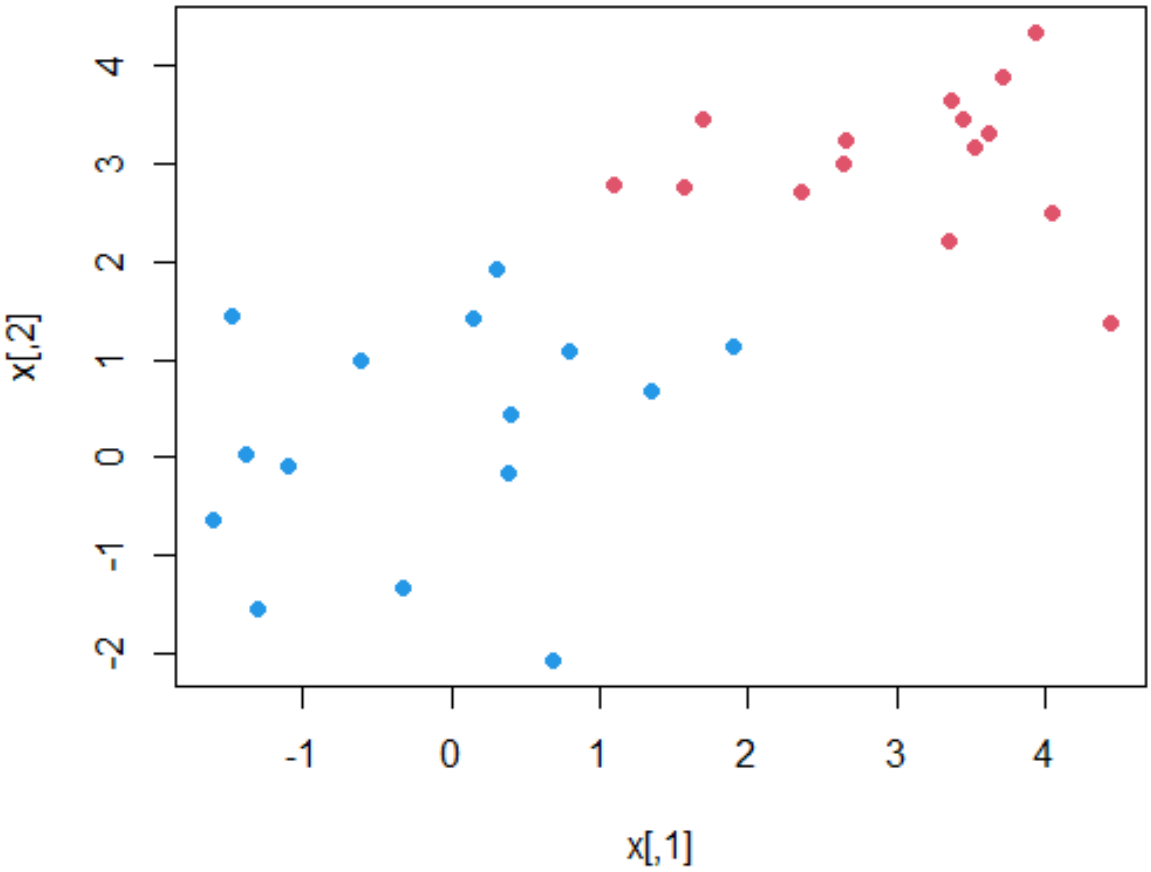



Figure 12: Imagen8

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

Figure 13: Imagen9

$$1 + 2X_1 + 3X_2 = 0$$

Figure 14: Imagen10

$$1 + 2X_1 + 3X_2 = 0$$

Figure 15: Imagen11

Si $\beta_0, \beta_1, \cdots, \beta_p$ son los coeficientes del hiperplano de margen máximo, entonces el clasificador de margen máximo clasifica la observación de prueba $x^*$ basado en el signo de

Figure 16: Imagen12

$$1 + 2X_1 + 3X_2 = 0$$

Figure 17: Imagen13

Si $\beta_0, \beta_1, \cdots, \beta_p$ son los coeficientes del hiperplano de margen máximo, entonces el clasificador de margen máximo clasifica la observación de prueba $x^*$ basado en el signo de

Figure 18: Imagen14

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$$

Figure 19: Imagen15

$$\max_{\beta_0,\beta_1,\cdots,\beta_p,M} M$$

Figure 20: Imagen16

$$\sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \cdots, n.$$

Figure 21: Imagen17

incluso de orden superior de los predictores. Por ejemplo, en vez de ajustar un clasificador de vectores de soporte usando p características

podríamos ajustar un clasificador de vectores de soporte usando 2p características

Entonces el problema de optimización

sujeto a

Se convertiría en:

sujeto a

No es difícil ver que hay muchas maneras de ampliar el espacio de características, y que a menos que seamos cuidadosos, podríamos terminar con un número enorme de características. Entonces los cálculos serían inmanejables.

## LA MAQUINA DE VECTORES DE SOPORTE

La máquina de vectores de soporte (SVM por sus siglas en inglés) es una extensión del clasificador de vectores de soporte que resulta de ampliar el espacio de características de una manera específica, usando kernels. Podemos querer ampliar nuestro espacio de características para acomodar una frontera no-lineal entre las clases. El enfoque del kernel que describimos aquí es simplemente un enfoque computacional eficiente para llevar a cabo esta idea.

PUEDE DEMOSTRARSE QUE **1**. El clasificador de vectores de soporte lineal se puede representar como:

donde hay $n$ parametros $i$,$i = 1, ..., n$, uno por observacion de entrenamiento.

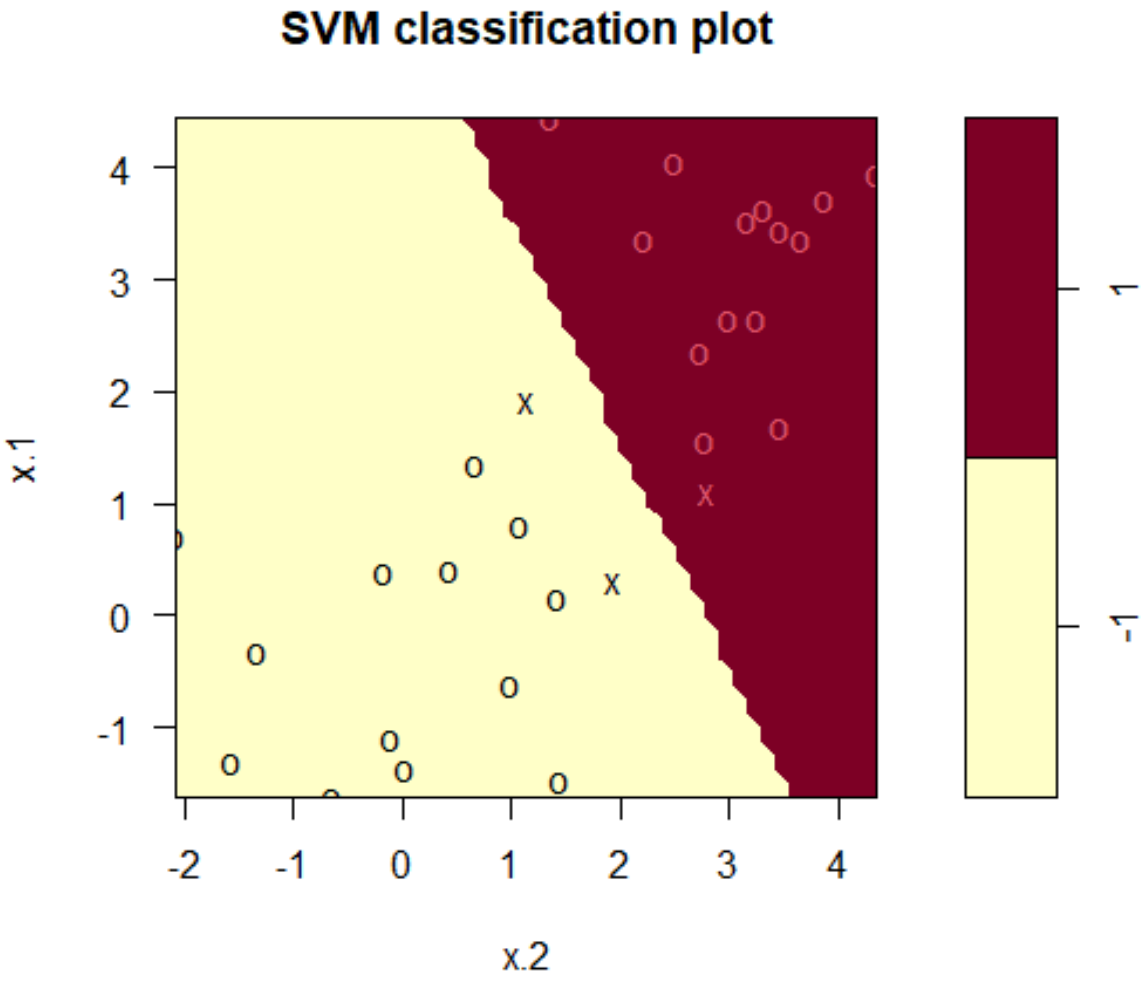



Figure 22: Imagen18

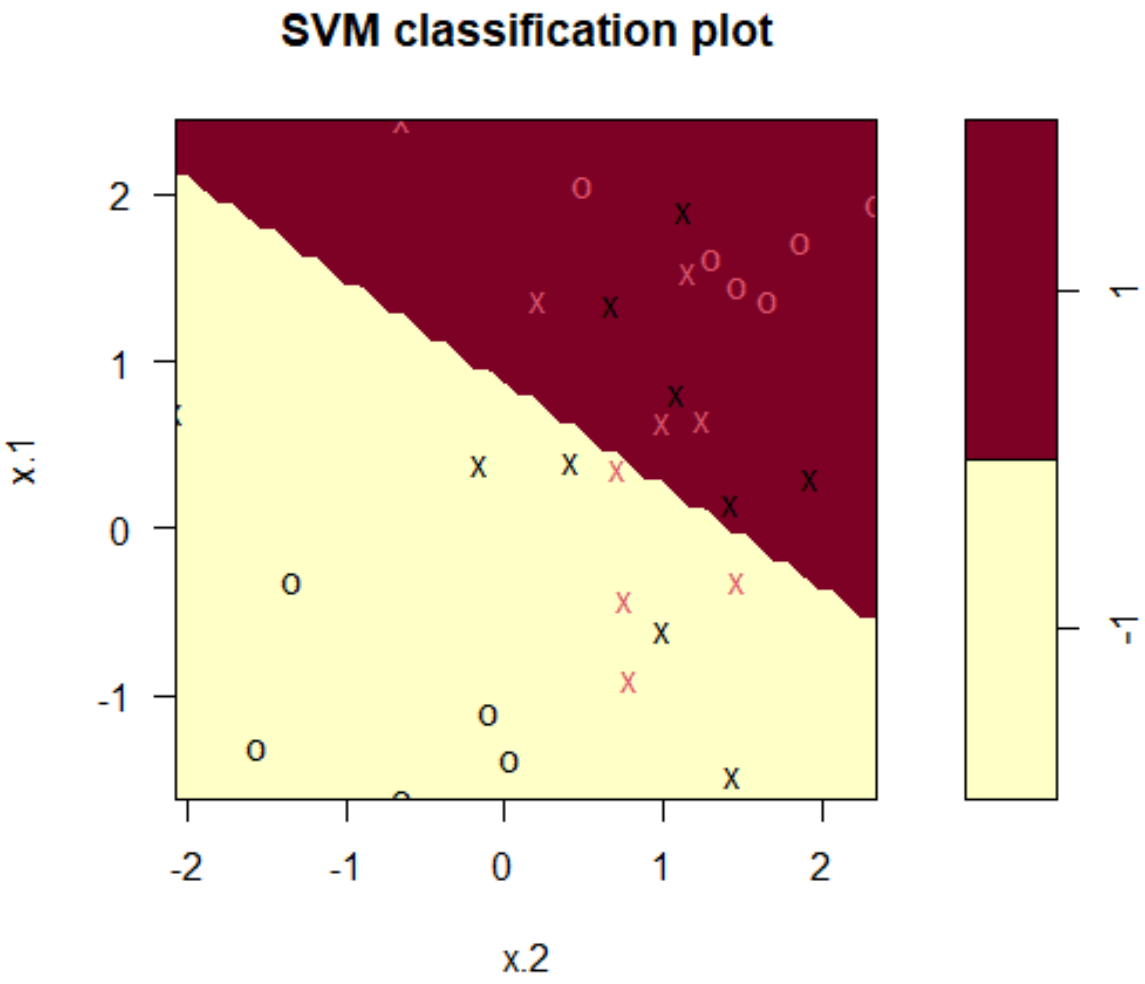



Figure 23: Imagen19

$$\max_{\beta_0,\beta_1,\cdots,\beta_p,\varepsilon_1,\cdots,\varepsilon_n,M} M$$

Figure 24: Imagen20

$$\sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \varepsilon_i),$$

$$\varepsilon_i \geq 0, \quad \sum_{i=1}^{n} \varepsilon_i \leq C,$$

Figure 25: Imagen21

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$$

Figure 26: Imagen22

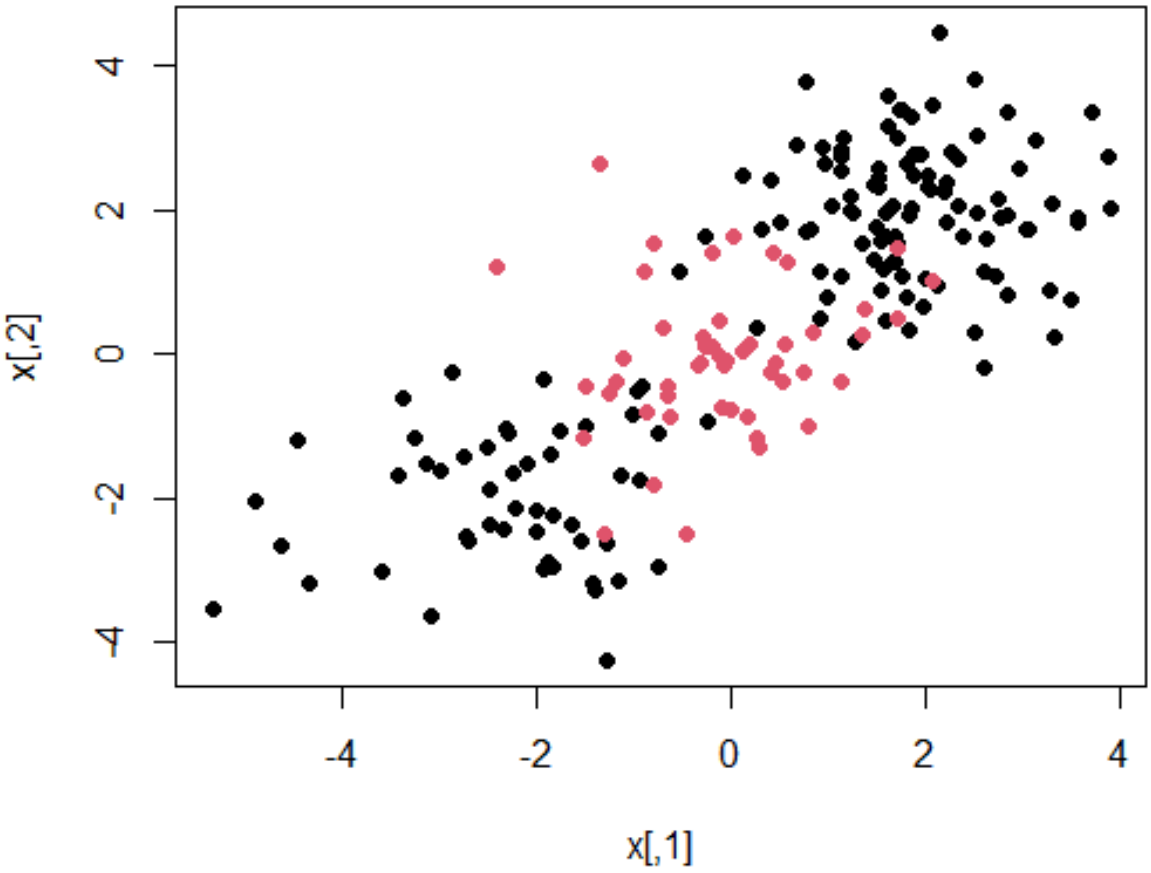



Figure 27: Imagen23

$$X_1, X_2, \cdots, X_p,$$

Figure 28: Imagen

$$X_1, X_1^2, X_2, X_2^2, \cdots, X_p, X_p^2.$$

Figure 29: Imagen

$$\max_{\beta_0,\beta_1,\cdots,\beta_p,\varepsilon_1,\cdots,\varepsilon_n,M} M$$

Figure 30: Imagen

$$\sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \varepsilon_i),$$

$$\varepsilon_i \geq 0, \quad \sum_{i=1}^{n} \varepsilon_i \leq C,$$

Figure 31: Imagen

$$\max_{\beta_0,\beta_{11},\beta_{12},\cdots,\beta_{p1},\beta_{p2},\varepsilon_1,\cdots,\varepsilon_n,M} M$$

Figure 32: Imagen

$$\sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1,$$

$$y_i(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2) \geq M(1-\varepsilon_i),$$

$$\varepsilon_i \geq 0, \quad \sum_{i=1}^{n}\varepsilon_i \leq C.$$

Figure 33: Imagen

El producto interno de dos observaciones $x_i$ , $x_{i'}$ está dado por

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}.$$

Figure 34: Imagen

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle,$$

Figure 35: Imagen

donde hay $n$ parámetros $\alpha_i$ , $i = 1, \cdots, n,$ uno por observación de entrenamiento.

2. Para estimar los parámetros $\alpha_1, \cdots, \alpha_n$ y $\beta_0$ , todo lo que necesitamos son los $\binom{n}{2}$ productos internos $\langle x_i, x_{i'} \rangle$ entre todos los pares de observaciones de entrenamiento.

Resulta que $\alpha_i$ es diferente de cero sólo para los vectores de soporte en la solución-es decir, si una observación de entrenamiento no es un vector de soporte, entonces su $\alpha_i$ es igual a cero-.

Si $S$ es la colección de índices de estos puntos de soporte, podemos re-escribir cualquier función de solución como

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

Figure 36: Imagen

# KERNEL

Un kernel es la funcion que cuantifica la similaridad de dos observaciones.

## KERNEL LINEAL

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j},$$

Figure 37: Imagen

## KERNEL POLINOMIAL

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d.$$

Figure 38: Imagen

Cuando el clasificador de vectores de soporte se combina con un kernel no-lineal, el clasificador que resulta se conoce como una máquina de vectores de soporte. En este caso la función tiene la forma

## KERNEL RADIAL

Comportamiento local del kernel radial

## CLASIFICACIÓN CON FRONTERA DE DECISION NO LINEAL

¿Cuál es la ventaja de usar un kernel en lugar de simplemente ampliar el espacio de características usando funciones de las características originales?

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i).$$

Figure 39: Imagen

$$K(x_i, x_{i'}) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

Figure 40: Imagen

$$K(x^*, x_h) = exp(-\gamma \sum_{j=1}^{p} (x_j^* - x_{hj})^2).$$

$$f(x^*) = \beta_0 + \sum_{i \in S} \alpha_i K(x^*, x_i).$$

Figure 41: Imagen

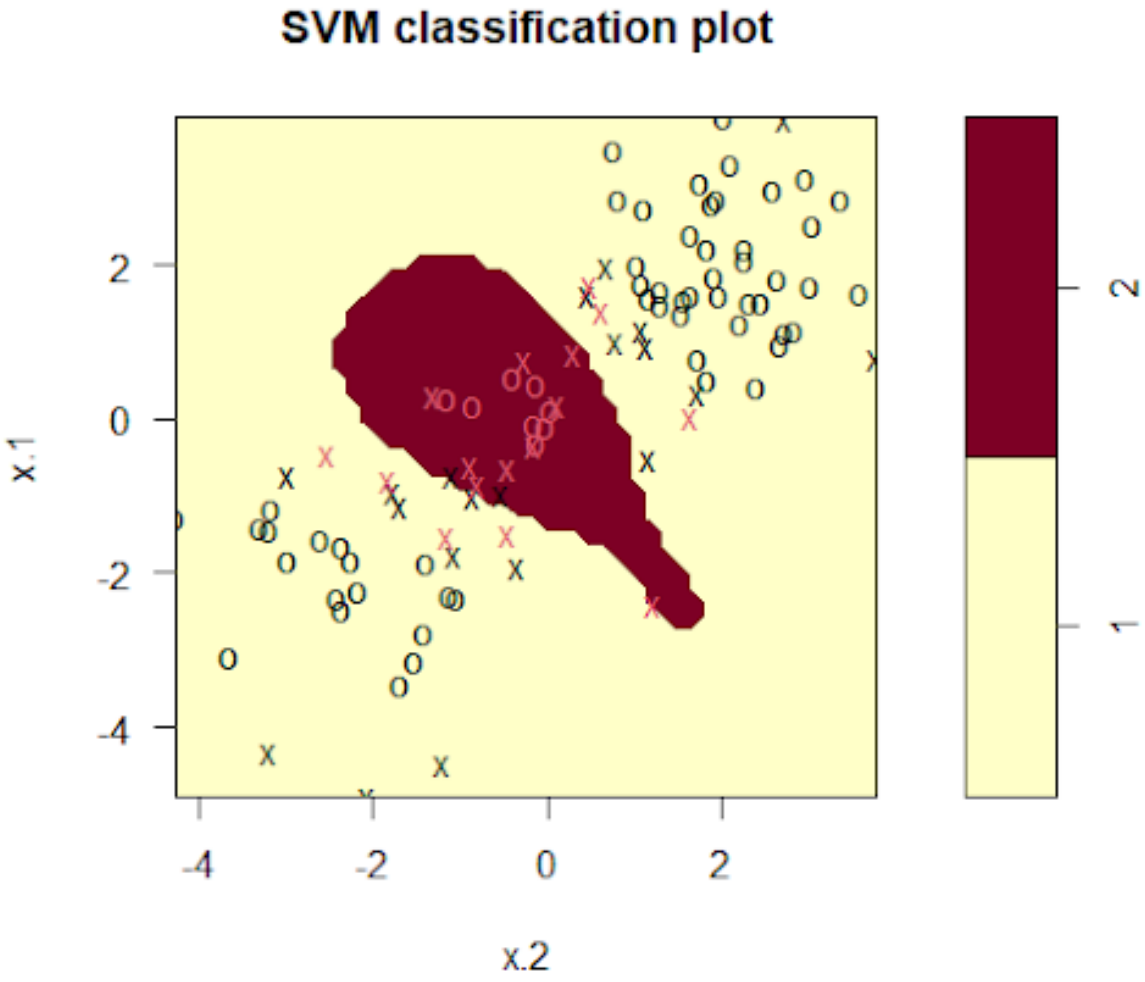



Figure 42: Imagen

Sólo necesitamos calcular $K(x_i, x_{i'})$ para todos los pares distintos $i$, $i'$.

Figure 43: Imagen

# EJEMPLO 1. REGRESION LINEAL SIMPLE

## OBJETIVO

Predecir una variable numérica a partir de otra variable predictora, cuando exista una relación aproximadamente lineal entre las variables y sea razonable asumir algunos supuestos.

## REQUISITOS

- Tener instalado R y RStudio
- Haber estudiado el Prework

```
# Ejemplo 1. Regresión Lineal Simple

# Primero hay que establecer el directorio de trabajo y este deberá contener
# el archivo de datos production.txt

# Leemos nuestros datos con la función read.table

production <- read.table("production.txt", header = TRUE)

# Los datos que importamos a R se encuentran como data frame con nombre
# production. Aplicamos la función attach al data frame production para
# poder manipular las columnas mediante sus nombres

attach(production)

# Hacemos el gráfico de dispersión

plot(RunSize, RunTime, xlab = "Tamaño de ejecución",
     ylab = "Tiempo de ejecución", pch = 16)

# Ajustamos un modelo de regresión lineal simple con la función lm, en donde
# la variable de respuesta es RunTime y la variable predictora es RunSize.
# Guardamos nuestro modelo ajustado con el nombre de m1

m1 <- lm(RunTime~RunSize)

# Obtenemos un resumen de nuestro modelo ajustado mediante la función 'summary'

summary(m1)
```

```
Call:
lm(formula = RunTime ~ RunSize)

Residuals:
    Min      1Q  Median      3Q     Max
-28.597 -11.079   3.329   8.302  29.627

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) 149.74770    8.32815   17.98 6.00e-13 ***
RunSize       0.25924    0.03714    6.98 1.61e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 16.25 on 18 degrees of freedom
Multiple R-squared:  0.7302,    Adjusted R-squared:  0.7152
F-statistic: 48.72 on 1 and 18 DF,  p-value: 1.615e-06
```

```r
# Graficamos nuestros datos nuevamente, pero ahora con la recta de regresión
# ajustada

plot(RunSize, RunTime, xlab = "Tamaño de ejecución",
     ylab = "Tiempo de ejecución", pch = 16)
abline(lsfit(RunSize, RunTime)) # Trazamos la recta de regresión estimada
mtext(expression(paste('Modelo de regresión lineal simple:',
                       ' ',
                       y[i] == beta[0] + beta[1]*x[i] + e[i])),
      side = 3, adj=1, font = 2)

# Recta de regresión poblacional

text(x = 200, y = 240, expression(paste('Recta de regresión:',
                                        ' ',
                                        y[i] == beta[0] + beta[1]*x[i])),
     adj = 1, font = 2)


# Recta de regresión estimada

text(x = 350, y = 180, expression(paste('Recta estimada:',
                                        ' ',
                                        hat(y)[i] == hat(beta)[0] + hat(beta)[1]*x[i])),
     adj = 1, font = 2)

# Recta de regresión estimada

text(x = 350, y = 160, expression(paste('Recta estimada:',
                                        ' ',
                                        hat(y)[i] == 149.74770 + 0.25924*x[i])),
     adj = 1, font = 2)

# Residuales

points(189, 215, pch=16, col = "red") # Punto muestral
149.74770 + 0.25924 * 189 # Valor y sobre la recta estimada
```

```
[1] 198.7441
```

```r
lines(c(189, 189), c(198.7441, 215), col = "red")

points(173, 166, pch=16, col = "red") # Punto muestral
149.74770 + 0.25924 * 173 # Valor y sobre la recta estimada
```

```
[1] 194.5962
```

```
lines(c(173, 173), c(166, 194.5962), col = "red")

# Acontinuación encontramos el cuantil de orden 0.975 de la distribución
# t de Student con 18 (n - 2) grados de libertad. En total tenemos n = 20
# observaciones en nuestro conjunto de datos. Estamos encontrando el valor
# que satisface P(T > tval) = 0.025

tval <- qt(1-0.05/2, 18)
tval
```

```
[1] 2.100922
```

```
# Comprobamos

pt(tval, df = 18)
```

```
[1] 0.975
```

```
# Encontramos intervalos de confianza del 95% para el intercepto y la pendiente
# del modelo de regresión lineal simple

round(confint(m1, level = 0.95), 3)
```

```
             2.5 %  97.5 %
(Intercept) 132.251 167.244
RunSize       0.181   0.337
```

```
# Ahora encontramos intervalos de confianza del 95% para la recta de regresión
# poblacional en algunos valores de X (RunSize)

RunSize0 <- c(50,100,150,200,250,300,350) # Algunos posibles valores de RunSize

(conf <- predict(m1, newdata =
                   data.frame(RunSize = RunSize0),
                 interval = "confidence", level = 0.95))
```

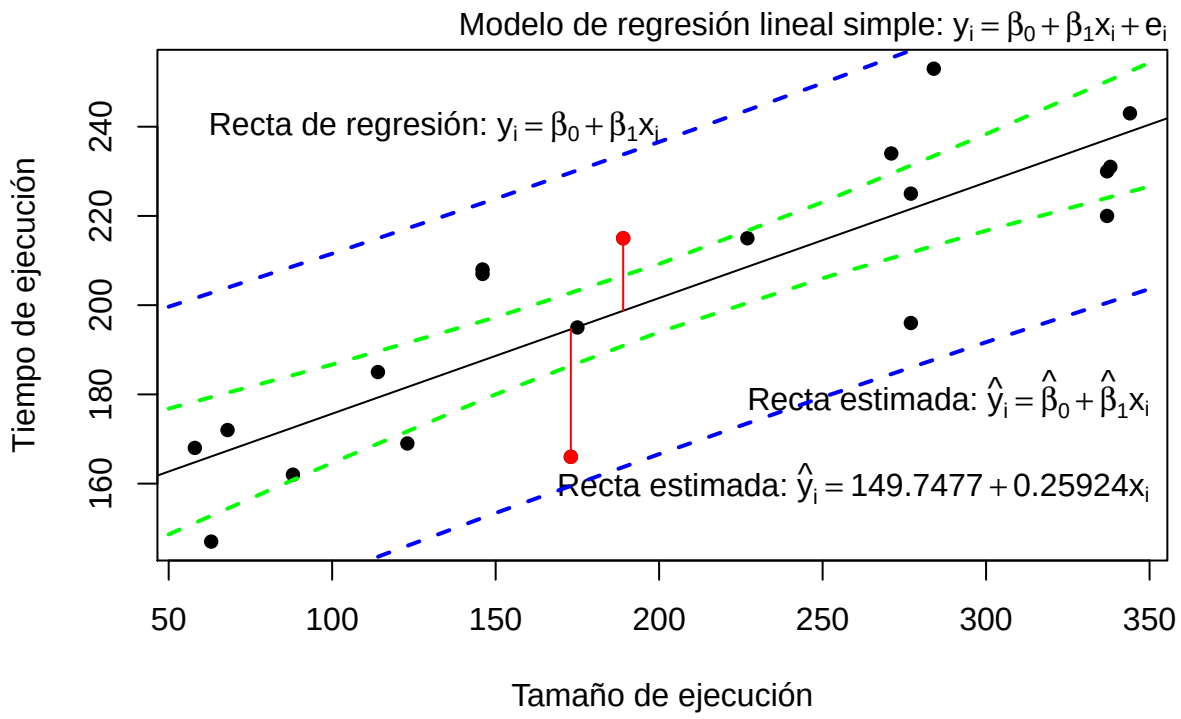
```
       fit      lwr      upr
1 162.7099 148.6204 176.7994
2 175.6720 164.6568 186.6872
3 188.6342 179.9969 197.2714
4 201.5963 193.9600 209.2326
5 214.5585 206.0455 223.0714
6 227.5206 216.7006 238.3407
7 240.4828 226.6220 254.3435
```

```
# Podemos visualizar gráficamente estos intervalos de confianza

lines(RunSize0, conf[, 2], lty = 2, lwd = 2, col = "green") # límites inferiores
lines(RunSize0, conf[, 3], lty = 2, lwd = 2, col = "green") # límites superiores
```

```r
# También podemos encontrar intervalos de predicción del 95% para el valor
# real de la variable de respuesta Y (RunTime) en algunos valores de X (RunSize)

(pred <- predict(m1, newdata =
          data.frame(RunSize = RunSize0),
        interval = "prediction", level = 0.95))
```

```
       fit      lwr      upr
1 162.7099 125.7720 199.6478
2 175.6720 139.7940 211.5500
3 188.6342 153.4135 223.8548
4 201.5963 166.6076 236.5850
5 214.5585 179.3681 249.7489
6 227.5206 191.7021 263.3392
7 240.4828 203.6315 277.3340
```

```r
# Podemos visualizar gráficamente estos intervalos de predicción

lines(RunSize0, pred[, 2], lty = 2, lwd = 2, col = "blue") # límites inferiores
lines(RunSize0, pred[, 3], lty = 2, lwd = 2, col = "blue") # límites superiores
```

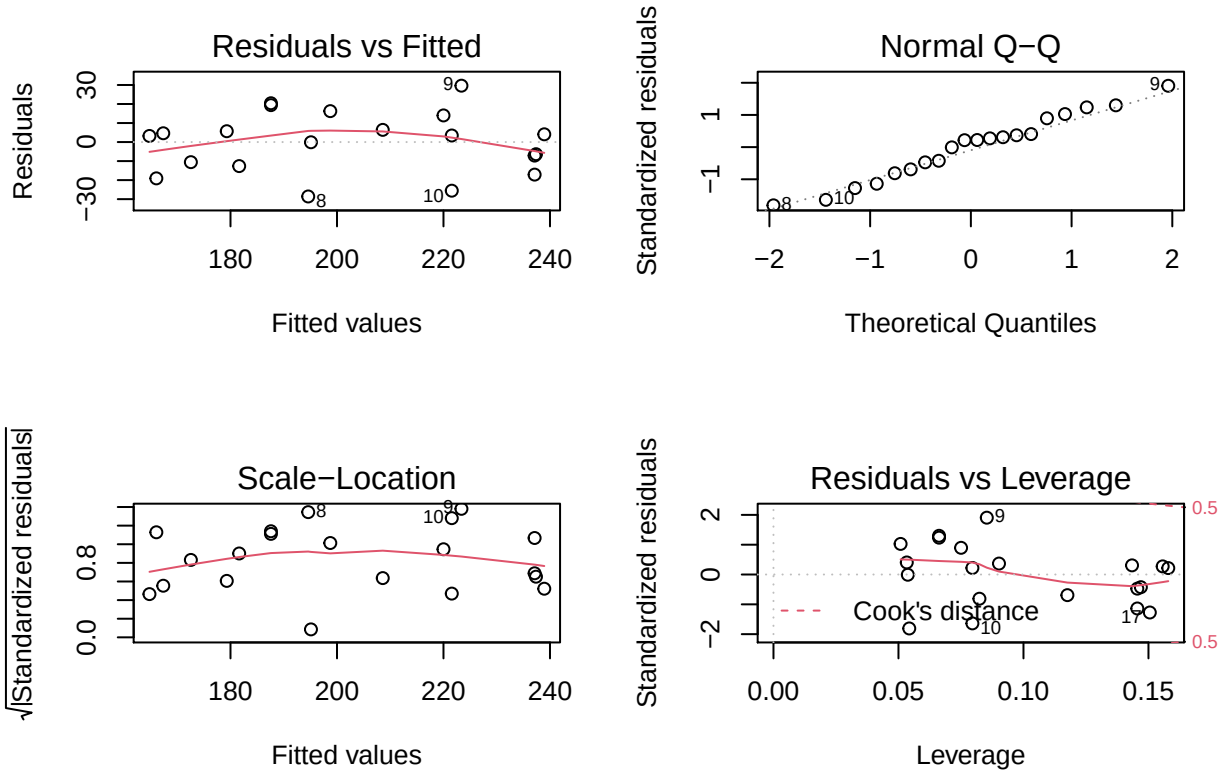



```r
# Note como los intervalos de confianza están contenidos dentro de los
# intervalos de predicción correspondientes
```

```r
# También es posible llevar a cabo un análisis de varianza para decidir si
# existe asociación lineal entre RunSize y RunTime

anova(m1)
```

```
Analysis of Variance Table

Response: RunTime
          Df  Sum Sq Mean Sq F value    Pr(>F)
RunSize    1 12868.4 12868.4  48.717 1.615e-06 ***
Residuals 18  4754.6   264.1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Gráfico de diagnóstico de R

# Cuando usamos un modelo de regresión, hacemos una serie de suposiciones.
# Entonces debemos hacer diagnósticos de regresión para verificar las
# supocisiones.

par(mfrow = c(2, 2))
plot(m1)
```



```r
dev.off()
```

```
null device
          1
```

```r
# Inspirado en:

# [S.J. Sheather, A Modern Approach to Regression with R, DOI: 10.1007/978-0-387-09608-7_2, © Springer
```

# RETO 1 REGRESION LINEAL SIMPLE

## OBJETIVO

- Ajustar un modelo de regresión lineal simple a una muestra de datos, cuando parezca que existe una relación lineal entre las variables subyacentes
- Obtener gráficas de diagnóstico, para decidir si es razonable asumir algunos supuestos necesarios para el modelo de regresión lineal simple

## DESARROLLO

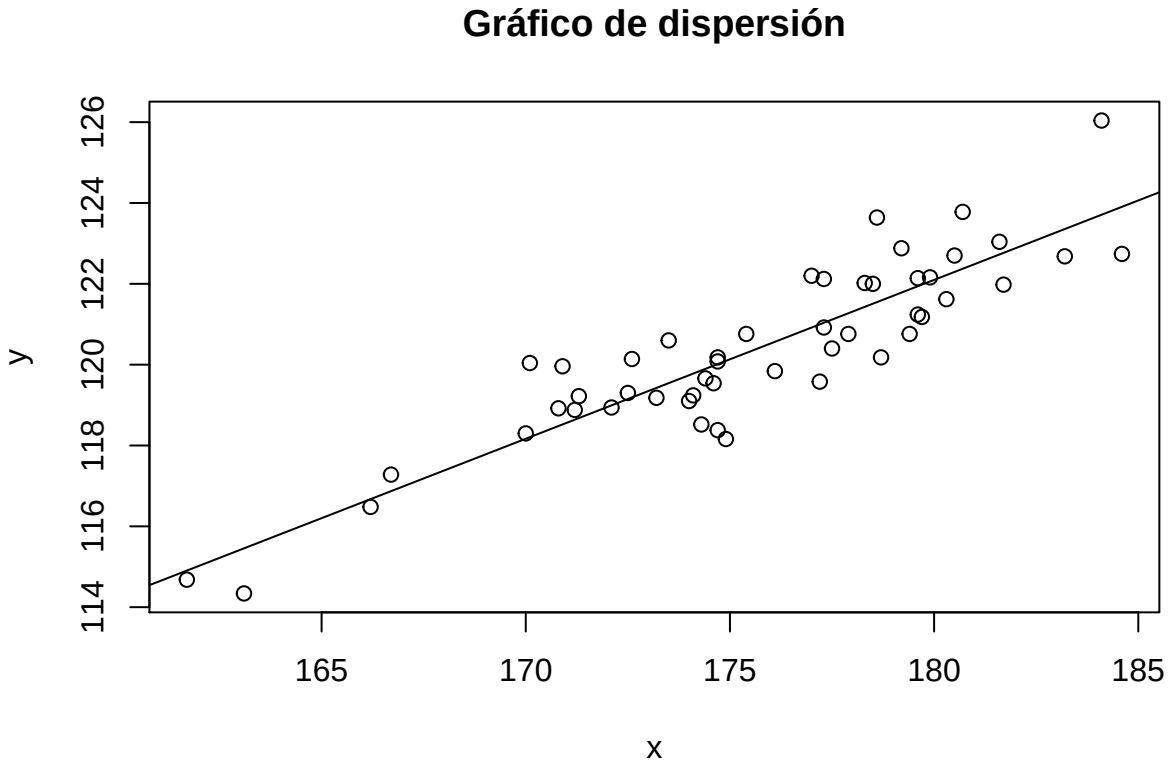
Se cree que entre las variables $x$ y $y$ del archivo csv adjunto, podría haber una relación más o menos lineal. Para tener más evidencia sobre esto lleve a cabo lo siguiente:

1. Realice el gráfico de dispersión de los datos
2. Ajuste un modelo de regresión lineal simple a los datos, muestre un resumen del modelo ajustado y trace la recta de regresión estimada junto con el gráfico de dispersión
3. Obtenga algunas gráficas de diagnóstico y diga si es razonable suponer para los errores aleatoriedad, normalidad y varianza constante.

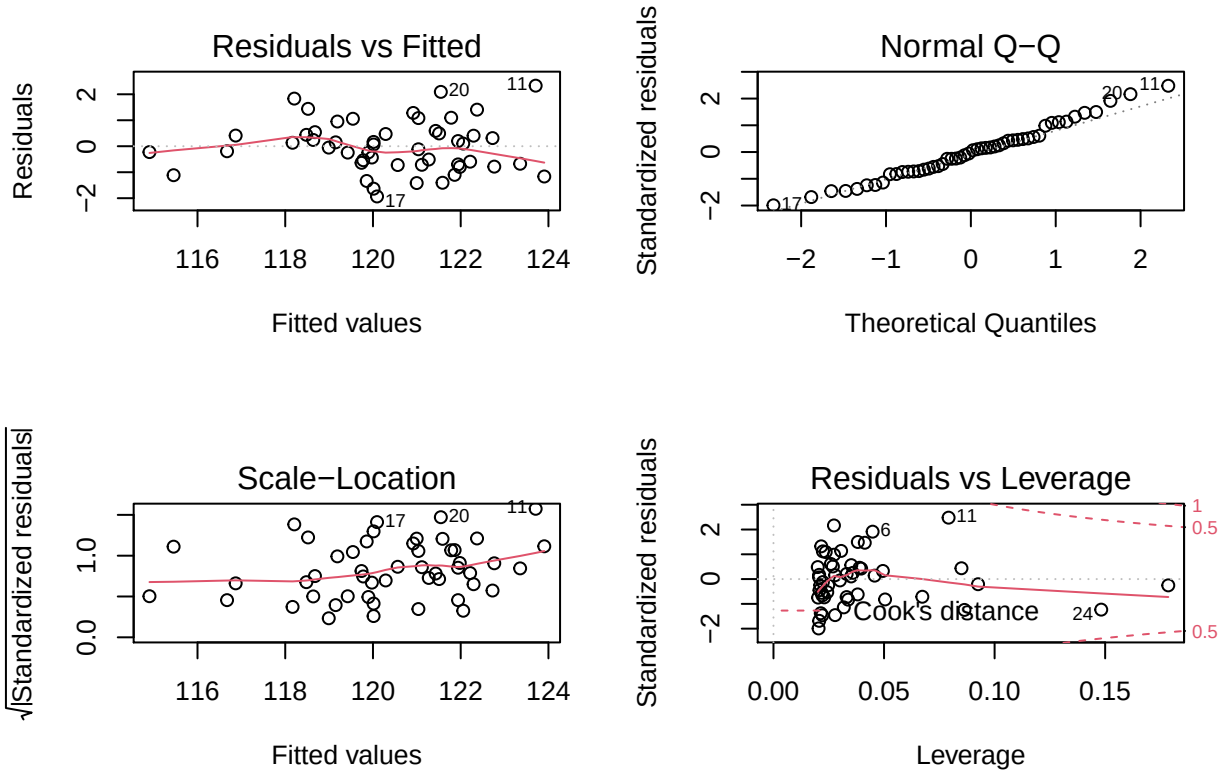
```r
# Reto 1. Regresión lineal simple

# Se cree que entre las variables x y y del archivo csv adjunto, podría haber una relación más o menos

# 1. Realice el gráfico de dispersión de los datos
# 2. Ajuste un modelo de regresión lineal simple a los datos, muestre un resumen del modelo ajustado y
# 3. Obtenga algunas gráficas de diagnóstico y diga si es razonable suponer para los errores aleatoried

# **Solución**

# Establezca primero un directorio de trabajo donde
# deberán estar los datos a importar
rm(list = ls()) # Para eliminar objetos creados previamente
datos <- read.csv("datos.csv")
attach(datos)
plot(x, y, main = "Gráfico de dispersión") # 1

modelo <- lm(y ~ x) # 2.
summary(modelo)
```

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-1.93292 -0.69381  0.00661  0.48681  2.33133

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 51.35438    4.93251   10.41 6.66e-14 ***
x            0.39302    0.02808   14.00  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9804 on 48 degrees of freedom
Multiple R-squared:  0.8032,    Adjusted R-squared:  0.7991
F-statistic: 195.9 on 1 and 48 DF,  p-value: < 2.2e-16
```

```r
abline(lsfit(x, y))
```




```r
par(mfrow = c(2, 2))
plot(modelo) # 3.
```




Primero hay que establecer el directorio de trabajo y este deberá contener el archivo de datos production.txt

Leemos nuestros datos con la función read.table

production <- read.table("production.txt", header = TRUE) Los datos que importamos a R se encuentran como data frame con nombre production. Aplicamos la función attach al data frame production para poder manipular las columnas mediante sus nombres.

attach(production) Hacemos el gráfico de dispersión

plot(RunSize, RunTime, xlab = "Tamaño de ejecución", ylab = "Tiempo de ejecución", pch = 16) Ajustamos un modelo de regresión lineal simple con la función lm, en donde la variable de respuesta es RunTime y la variable predictora es RunSize. Guardamos nuestro modelo ajustado con el nombre de m1.

m1 <- lm(RunTime~RunSize) Obtenemos un resumen de nuestro modelo ajustado mediante la función summary

summary(m1) Graficamos nuestros datos nuevamente, ahora con la recta de regresión estimada, mostrando algunas ecuaciones y algunos residuales gráficamente.

plot(RunSize, RunTime, xlab = "Tamaño de ejecución", ylab = "Tiempo de ejecución", pch = 16) abline(lsfit(RunSize, RunTime)) # Trazamos la recta de regresión estimada mtext(expression(paste('Modelo de regresión lineal simple:', ' ', y[i] == beta[0] + beta[1]*x[i] + e[i])), side = 3, adj=1, font = 2)

## Recta de regresión poblacional

text(x = 200, y = 240, expression(paste('Recta de regresión:', ' ', y[i] == beta[0] + beta[1]*x[i])), adj = 1, font = 2)

## Recta de regresión estimada

text(x = 350, y = 180, expression(paste('Recta estimada:', ' ', hat(y)[i] == hat(beta)[0] + hat(beta)[1]*x[i])), adj = 1, font = 2)

## Recta de regresión estimada

text(x = 350, y = 160, expression(paste('Recta estimada:', ' ', hat(y)[i] == 149.74770 + 0.25924*x[i])), adj = 1, font = 2)

## Residuales

points(189, 215, pch=16, col = "red") # Punto muestral 149.74770 + 0.25924 * 189 # Valor y sobre la recta estimada lines(c(189, 189), c(198.7441, 215), col = "red")

points(173, 166, pch=16, col = "red") # Punto muestral 149.74770 + 0.25924 * 173 # Valor y sobre la recta estimada lines(c(173, 173), c(166, 194.5962), col = "red") Acontinuación encontramos el cuantil de orden 0.975 de la distribución t de Student con 18 (n - 2) grados de libertad. En total tenemos n = 20 observaciones en nuestro conjunto de datos. Estamos encontrando el valor que satisface P(T > tval) = 0.025

tval <- qt(1-0.05/2, 18) tval Comprobamos

pt(tval, df = 18) Encontramos intervalos de confianza del 95% para el intercepto y la pendiente del modelo de regresión lineal simple

round(confint(m1, level = 0.95), 3) Ahora encontramos intervalos de confianza del 95% para la recta de regresión poblacional en algunos valores de X (RunSize)

RunSize0 <- c(50,100,150,200,250,300,350) # Algunos posibles valores de RunSize

(conf <- predict(m1, newdata = data.frame(RunSize = RunSize0), interval = "confidence", level = 0.95))

## Podemos visualizar gráficamente estos intervalos de confianza

lines(RunSize0, conf[, 2], lty = 2, lwd = 2, col = "green") # límites inferiores lines(RunSize0, conf[, 3], lty = 2, lwd = 2, col = "green") # límites superiores También podemos encontrar intervalos de predicción del 95% para el valor real de la variable de respuesta Y (RunTime) en algunos valores de X (RunSize)

(pred <- predict(m1, newdata = data.frame(RunSize = RunSize0), interval = "prediction", level = 0.95))

## Podemos visualizar gráficamente estos intervalos de predicción

lines(RunSize0, pred[, 2], lty = 2, lwd = 2, col = "blue") # límites inferiores lines(RunSize0, pred[, 3], lty = 2, lwd = 2, col = "blue") # límites superiores Note como los intervalos de confianza están contenidos dentro de los intervalos de predicción correspondientes.

También es posible llevar a cabo un análisis de varianza para decidir si existe asociación lineal entre RunSize y RunTime

anova(m1) Gráficos de diagnóstico de R Cuando usamos un modelo de regresión, hacemos una serie de suposiciones. Entonces debemos hacer diagnósticos de regresión para verificar las supocisiones.

par(mfrow = c(2, 2)) plot(m1) dev.off()

# EJEMPLO 2. REGRESIÓN LINEAL MULTIPLE

## OBJETIVO

- Aprender como en ocasiones es posible predecir una variable numérica a partir de otras variables predictoras cuando exista una relación lineal entre las variables y sea razonable asumir algunos supuestos.

## DESARROLLO

Supongamos que queremos emprender un negocio o que se nos colicita un estudio en en cual se requiere predecir el precio de cena (platillo), para poder estar dentro de los rangos de precios del mercado y que el restaurante sea rentable.

Entonces primero vamos a analizar los datos de encuestas de clientes de 168 restaurantes Italianos en el área deseada que están disponibles, los cuales tienen las siguientes variables de estudio: * Y: Price (Precio): el precio (en USD) de la cena * X1: Food: Valuación del cliente de la comida (sacado de 30) * X2: Décor: Valuación del cliente de la decoración (sacado de 30) * X3: Service: Valuación del cliente del servicio (sacado de 30) * X4: East: variable dummy: 1 (0) si el restaurante está al este (oeste) de la quinta avenida.

Primero debemos establecer nuestro directorio de trabajo y el archivo de datos (nyc.csv) que importaremos a R deberá de estar en este directorio.

nyc <- read.csv("nyc.csv", header = TRUE) Observamos algunas filas y la dimensión del data frame

head(nyc, 2); tail(nyc, 2); dim(nyc) attach(nyc) Llevamos a cabo el ajuste de un modelo Y = beta0 + beta1Food + beta2Decor + beta3Service + beta4East + e

m1 <- lm(Price ~ Food + Decor + Service + East) Obtenemos un resumen

summary(m1) Ajustamos nuevamente un modelo pero ahora sin considerar la variable Service ya que en el resultado anterior se observó que su coeficiente de regresión no fue estadísticamente significativo Y = beta0 + beta1Food + beta2Decor + beta4*East + e (Reducido)

m2 <- lm(Price ~ Food + Decor + East) Obtenemos un resumen del modelo ajustado

summary(m2) Una forma alternativa de obtener m2 es usar el comando update

m2 <- update(m1, ~.-Service) summary(m2) Análisis de covarianza Para investigar si el efecto de los predictores depende de la variable dummy East consideraremos el siguiente modelo el cual es una extensión a más de una variable predictora del modelo de rectas de regresión no relacionadas Y = beta0 + beta1Food + beta2Decor + beta3Service + beta4East + beta5FoodEast + beta6DecorEast + beta7ServiceEast + e (Completo)

mfull <- lm(Price ~ Food + Decor + Service + East + Food:East + Decor:East + Service:East) Note como ninguno de los coeficientes de regresión para los términos de interacción son estadísticamente significativos

summary(mfull) Ahora compararemos el modelo completo guardado en mfull contra el modelo reducido guardado en m2. Es decir, llevaremos a cabo una prueba de hipótesis general de

H0: beta3 = beta5 = beta6 = beta7 = 0

es decir Y = beta0 + beta1Food + beta2Decor + beta4*East + e (Reducido)

contra

H1: H0 no es verdad

es decir, Y = beta0 + beta1Food + beta2Decor + beta3Service + beta4East + beta5FoodEast + beta6DecorEast + beta7ServiceEast + e (Completo)

La prueba de si el efecto de los predictores depende de la variable dummy East puede lograrse usando la siguiente prueba-F parcial.

anova(m2,mfull) Dado que el p-value es aproximadamente 0.36, fallamos en rechazar la hipótesis nula y adopatamos el modelo reducido Y = beta0 + beta1Food + beta2Decor + beta4*East + e (Reducido)

Diagnósticos En regresión múltiple, las gráficas de residuales o de residuales estandarizados proporcionan información directa sobre la forma en la cual el modelo está mal especificado cuando se cumplen las siguientes dos condiciones:

E(Y | X = x) = g(beta0 + beta1x1 + ... + betapxp)

y

E(Xi | Xj) aprox alpha0 + alpha1*Xj

Cuando estas condiciones se cumplen, la gráfica de Y contra los valores ajustados, proporciona información directa acerca de g. En regresión lineal múltiple g es la función identidad. En este caso la gráfica de Y contra los valores ajustados debe producir puntos dispersos alrededor de una recta. Si las condiciones no se cumplen, entonces un patrón en la gráfica de los residuales indica que un modelo incorrecto ha sido ajustado, pero el patrón mismo no proporciona información directa sobre como el modelo está mal específicado.

Ahora tratemos de verificar si el modelo ajustado es un modelo válido.

Acontinuación mostramos una matriz de gráficos de dispersión de los tres predictores continuos. Los predictores parecen estar linealmente relacionados al menos aproximadamente

pairs(~ Food + Decor + Service, data = nyc, gap = 0.4, cex.labels = 1.5) Acontinuación veremos gráficas de residuales estandarizados contra cada predictor. La naturaleza aleatoria de estas gráficas es un indicativo de que el modelo ajustado es un modelo válido para los datos.

m1 <- lm(Price ~ Food + Decor + Service + East) summary(m1) StanRes1 <- rstandard(m1) par(mfrow = c(2, 2)) plot(Food, StanRes1, ylab = "Residuales Estandarizados") plot(Decor, StanRes1, ylab = "Residuales Estandarizados") plot(Service, StanRes1, ylab = "Residuales Estandarizados") plot(East, StanRes1, ylab = "Residuales Estandarizados") dev.off() Finalmente mostramos una gráfica de Y, el precio contra los valores ajustados

plot(m1$fitted.values, Price, xlab = "Valoresajustados", ylab = "Price")abline(lsfit(m1fitted.values, Price))

```
# Ejemplo 2. Regresión Lineal Múltiple

# Predecir el precio de cena (platillo).
# Datos de encuestas de clientes de 168 restaurantes Italianos
# en el área deseada están disponibles.

# Y: Price (Precio): el precio (en USD) de la cena
# X1: Food: Valuación del cliente de la comida (sacado de 30)
# X2: Décor: Valuación del cliente de la decoración (sacado de 30)
# X3: Service: Valuación del cliente del servicio (sacado de 30)
# X4: East: variable dummy: 1 (0) si el restaurante está al este (oeste) de la quinta avenida

# Primero debemos establecer nuestro directorio de trabajo y el archivo
# de datos (nyc.csv) que importaremos a R deberá de estar en este directorio

nyc <- read.csv("nyc.csv", header = TRUE)

# Observamos algunas filas y la dimensión del data frame

head(nyc, 2); tail(nyc, 2); dim(nyc)
```

```
  Case          Restaurant Price Food Decor Service East
```

```
1    1 Daniella Ristorante    43   22    18      20    0
2    2  Tello's Ristorante    32   20    19      19    0

    Case Restaurant Price Food Decor Service East
167  167    Métisse    38   22    17      21    0
168  168    Gennaro    34   24    10      16    0

[1] 168   7
```

```r
attach(nyc)
```

```
The following object is masked from production:

    Case
```

```r
# Llevamos a cabo el ajuste de un modelo
# Y = beta0 + beta1*Food + beta2*Decor + beta3*Service + beta4*East + e

m1 <- lm(Price ~ Food + Decor + Service + East)

# Obtenemos un resumen

summary(m1)
```

```
Call:
lm(formula = Price ~ Food + Decor + Service + East)

Residuals:
     Min       1Q   Median       3Q      Max
-14.0465  -3.8837   0.0373   3.3942  17.7491

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.023800   4.708359  -5.102 9.24e-07 ***
Food          1.538120   0.368951   4.169 4.96e-05 ***
Decor         1.910087   0.217005   8.802 1.87e-15 ***
Service      -0.002727   0.396232  -0.007   0.9945
East          2.068050   0.946739   2.184   0.0304 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.738 on 163 degrees of freedom
Multiple R-squared:  0.6279,    Adjusted R-squared:  0.6187
F-statistic: 68.76 on 4 and 163 DF,  p-value: < 2.2e-16
```

```r
# Ajustamos nuevamente un modelo pero ahora sin considerar la variable Service
# ya que en el resultado anterior se observó que su coeficiente de regresión
# no fue estadísticamente significativo

# Y = beta0 + beta1*Food + beta2*Decor + beta4*East + e (Reducido)
```

```r
m2 <- lm(Price ~ Food + Decor + East)

# Obtenemos un resumen del modelo ajustado

summary(m2)
```

```
Call:
lm(formula = Price ~ Food + Decor + East)

Residuals:
     Min       1Q   Median       3Q      Max
-14.0451  -3.8809   0.0389   3.3918  17.7557

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.0269     4.6727  -5.142 7.67e-07 ***
Food          1.5363     0.2632   5.838 2.76e-08 ***
Decor         1.9094     0.1900  10.049  < 2e-16 ***
East          2.0670     0.9318   2.218   0.0279 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.72 on 164 degrees of freedom
Multiple R-squared:  0.6279,    Adjusted R-squared:  0.6211
F-statistic: 92.24 on 3 and 164 DF,  p-value: < 2.2e-16
```

```r
# Una forma alternativa de obtener m2 es usar el comando update

m2 <- update(m1, ~.-Service)
summary(m2)
```

```
Call:
lm(formula = Price ~ Food + Decor + East)

Residuals:
     Min       1Q   Median       3Q      Max
-14.0451  -3.8809   0.0389   3.3918  17.7557

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.0269     4.6727  -5.142 7.67e-07 ***
Food          1.5363     0.2632   5.838 2.76e-08 ***
Decor         1.9094     0.1900  10.049  < 2e-16 ***
East          2.0670     0.9318   2.218   0.0279 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.72 on 164 degrees of freedom
Multiple R-squared:  0.6279,    Adjusted R-squared:  0.6211
F-statistic: 92.24 on 3 and 164 DF,  p-value: < 2.2e-16
```

```
######

# Análisis de covarianza

# Para investigar si el efecto de los predictores depende de la variable dummy
# East consideraremos el siguiente modelo el cual es una extensión a más de una
# variable predictora del modelo de rectas de regresión no relacionadas
# Y = beta0 + beta1*Food + beta2*Decor +  beta3*Service + beta4*East
#           + beta5*Food*East + beta6*Decor*East + beta7*Service*East + e (Completo)

mfull <- lm(Price ~ Food + Decor + Service + East +
              Food:East + Decor:East + Service:East)

# Note como ninguno de los coeficientes de regresión para los
# términos de interacción son estadísticamente significativos

summary(mfull)
```

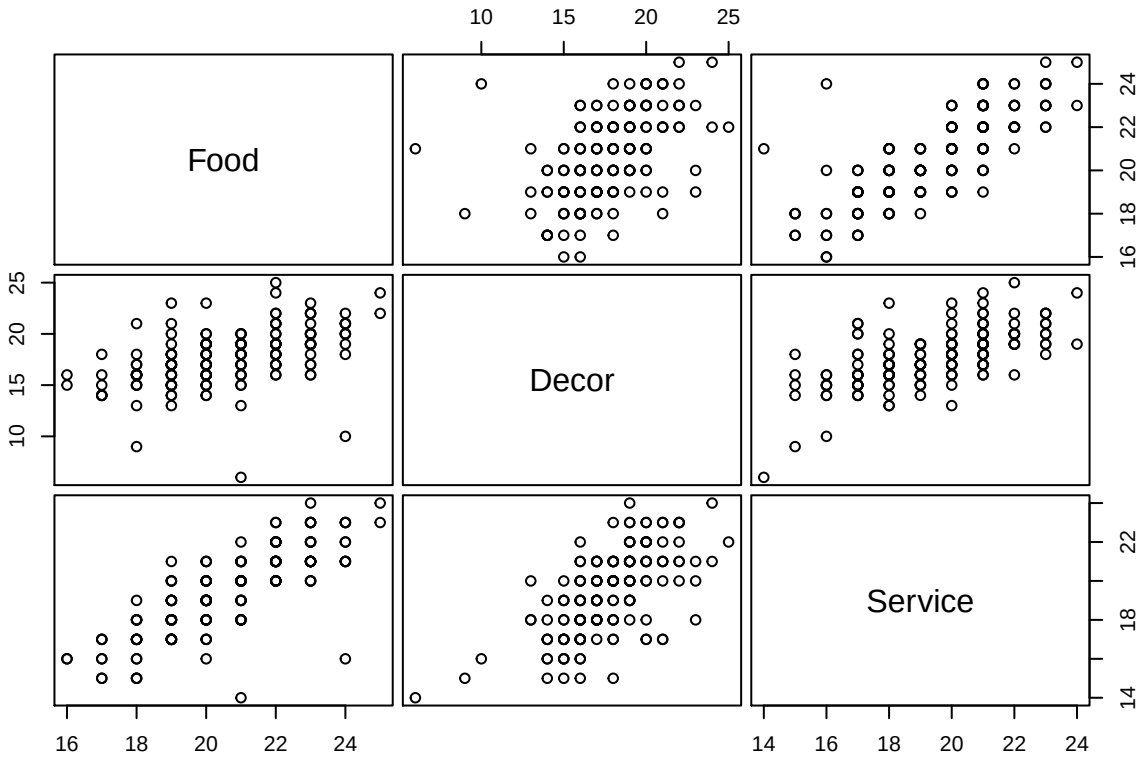
```
Call:
lm(formula = Price ~ Food + Decor + Service + East + Food:East +
    Decor:East + Service:East)

Residuals:
     Min       1Q   Median       3Q      Max
-13.5099  -3.7996  -0.1413   3.6522  17.1656

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -26.9949     8.4672  -3.188  0.00172 **
Food           1.0068     0.5704   1.765  0.07946 .
Decor          1.8881     0.2984   6.327  2.4e-09 ***
Service        0.7438     0.6443   1.155  0.25001
East           6.1253    10.2499   0.598  0.55095
Food:East      1.2077     0.7743   1.560  0.12079
Decor:East    -0.2500     0.4570  -0.547  0.58510
Service:East  -1.2719     0.8171  -1.557  0.12151
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.713 on 160 degrees of freedom
Multiple R-squared:  0.6379,    Adjusted R-squared:  0.622
F-statistic: 40.27 on 7 and 160 DF,  p-value: < 2.2e-16
```

```
# Ahora compararemos el modelo completo guardado en mfull contra el modelo
# reducido guardado en m2. Es decir, llevaremos a cabo una prueba de hipótesis
# general de

# H0: beta3 = beta5 = beta6 = beta7 = 0
# es decir Y = beta0 + beta1*Food + beta2*Decor + beta4*East + e (Reducido)
# contra
# H1: H0 no es verdad
# es decir,
```

```r
# Y = beta0 + beta1*Food + beta2*Decor +  beta3*Service + beta4*East
#           + beta5*Food*East + beta6*Decor*East + beta7*Service*East + e (Completo)

# La prueba de si el efecto de los predictores depende de la variable dummy
# East puede lograrse usando la siguiente prueba-F parcial.

anova(m2,mfull)
```

```
Analysis of Variance Table

Model 1: Price ~ Food + Decor + East
Model 2: Price ~ Food + Decor + Service + East + Food:East + Decor:East +
    Service:East
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    164 5366.5
2    160 5222.2  4    144.36 1.1057 0.3558
```

```r
# Dado que el p-value es aproximadamente 0.36, fallamos en rechazar la hipótesis
# nula y adopatamos el modelo reducido
# Y = beta0 + beta1*Food + beta2*Decor + beta4*East + e (Reducido)

######

# Diagnósticos

# En regresión múltiple, las gráficas de residuales o de residuales
# estandarizados proporcionan información directa sobre la forma
# en la cual el modelo está mal especificado cuando se cumplen
# las siguientes dos condiciones:

# E(Y | X = x) = g(beta0 + beta1*x1 + ... + betap*xp) y
# E(Xi | Xj) aprox alpha0 + alpha1*Xj

# Cuando estas condiciones se cumplen, la gráfica de Y contra
# los valores ajustados, proporciona información directa acerca de g.
# En regresión lineal múltiple g es la función identidad. En
# este caso la gráfica de Y contra los valores ajustados
# debe producir puntos dispersos alrededor de una recta.
# Si las condiciones no se cumplen, entonces un patrón en la
# gráfica de los residuales indica que un modelo incorrecto
# ha sido ajustado, pero el patrón mismo no proporciona
# información directa sobre como el modelo está mal específicado.

# Ahora tratemos de verificar si el modelo ajustado es un modelo válido.

# Acontinuación mostramos una matriz de gráficos de dispersión de los
# tres predictores continuos. Los predictores parecen estar linealmente
# relacionados al menos aproximadamente

pairs(~ Food + Decor + Service, data = nyc, gap = 0.4, cex.labels = 1.5)
```

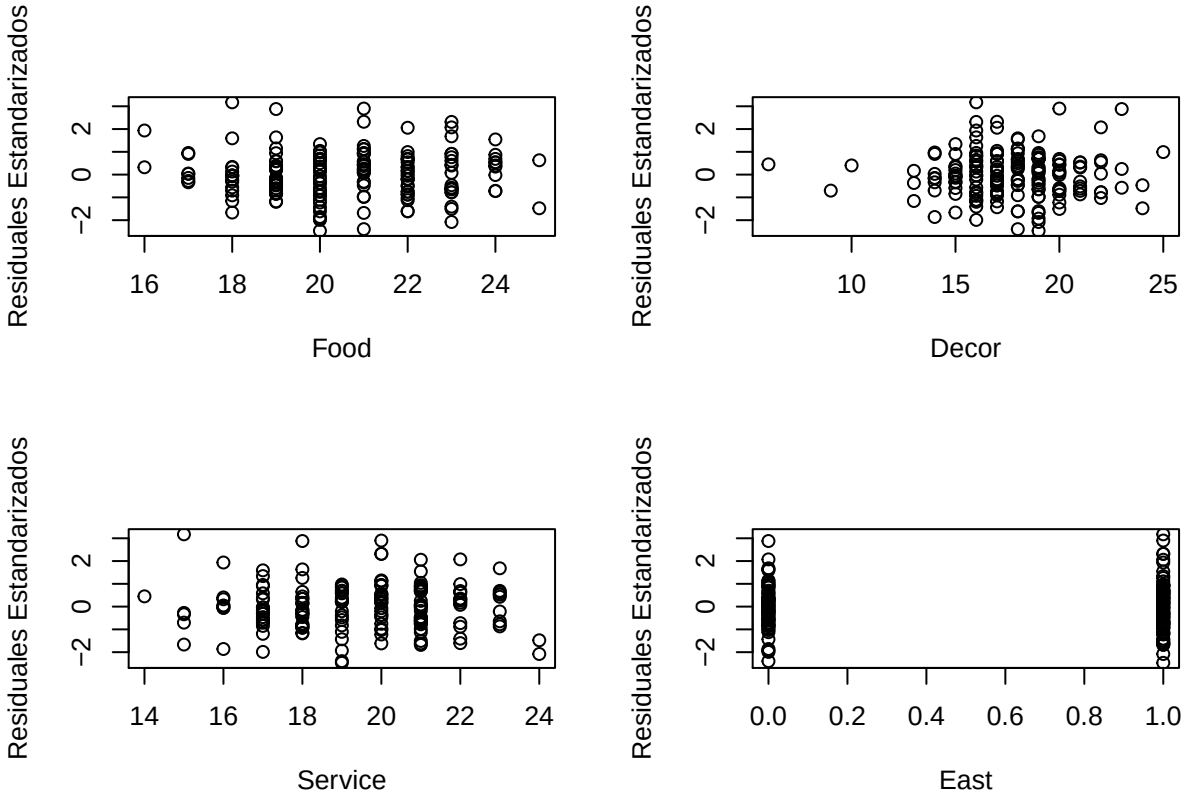



```r
# Acontinuación veremos gráficas de residuales estandarizados contra cada
# predictor. La naturaleza aleatoria de estas gráficas es un indicativo de
# que el modelo ajustado es un modelo válido para los datos.

m1 <- lm(Price ~ Food + Decor + Service + East)
summary(m1)
```

```
Call:
lm(formula = Price ~ Food + Decor + Service + East)

Residuals:
     Min       1Q   Median       3Q      Max
-14.0465  -3.8837   0.0373   3.3942  17.7491

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.023800   4.708359  -5.102 9.24e-07 ***
Food          1.538120   0.368951   4.169 4.96e-05 ***
Decor         1.910087   0.217005   8.802 1.87e-15 ***
Service      -0.002727   0.396232  -0.007   0.9945
East          2.068050   0.946739   2.184   0.0304 *
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 5.738 on 163 degrees of freedom
```

```
Multiple R-squared:  0.6279,    Adjusted R-squared:  0.6187
F-statistic: 68.76 on 4 and 163 DF,  p-value: < 2.2e-16
```

```r
StanRes1 <- rstandard(m1)
par(mfrow = c(2, 2))
plot(Food, StanRes1, ylab = "Residuales Estandarizados")
plot(Decor, StanRes1, ylab = "Residuales Estandarizados")
plot(Service, StanRes1, ylab = "Residuales Estandarizados")
plot(East, StanRes1, ylab = "Residuales Estandarizados")
```




```r
dev.off()
```

```
null device
          1
```

```r
# Finalmente mostramos una gráfica de Y, el precio contra los valores
# ajustados

plot(m1$fitted.values, Price, xlab = "Valores ajustados", ylab = "Price")
abline(lsfit(m1$fitted.values, Price))

# Inspirado en:

# [S.J. Sheather, A Modern Approach to Regression with R, DOI: 10.1007/978-0-387-09608-7_2, © Springer
```

# EJEMPLO 3 MAQUINAS DE VECTORES DE SOPORTE COMPAÑÍA DE TARJETAS DE CREDITO

## OBJETIVO

- Clasificar clientes potenciales de una compañía de tarjetas de crédito usando máquinas de vectores de soporte

## DESARROLLO

Paquetes de R utilizados

suppressMessages(suppressWarnings(library(dplyr))) suppressMessages(suppressWarnings(library(e1071))) suppressMessages(suppressWarnings(library(ggplot2))) suppressMessages(suppressWarnings(library(ISLR))) Observemos algunas características del data frame Default del paquete ISLR, con funciones tales como head, tail, dim y str. ?Default head(Default) tail(Default) dim(Default) str(Default) Usando ggplot del paquete ggplot2, realicemos un gráfico de dispersión con la variable balance en el eje x, la variable income en el eje y, diferenciando las distintas categorías en la variable default usando el argumento colour. Lo anterior para estudiantes y no estudiantes usando facet_wrap. ggplot(Default, aes(x = balance, y = income, colour = default)) + geom_point() + facet_wrap('student') + theme_grey() + ggtitle("Datos Default") Generemos un vector de índices llamado train, tomando de manera aleatoria 5000 números de los primeros 10,000 números naturales, esto servirá para filtrar el conjunto de entrenamiento y el conjunto de prueba del data frame Default. Realicemos el gráfico de dispersión análogo al punto 2, pero para los conjuntos de entrenamiento y de prueba. set.seed(2020) train = sample(nrow(Default), round(nrow(Default)/2)) tail(Default[train, ])

ggplot(Default[train, ], aes(x = balance, y = income, colour = default)) + geom_point() + facet_wrap('student') + theme_dark() + ggtitle("Conjunto de entrenamiento")

ggplot(Default[-train, ], aes(x = balance, y = income, colour = default)) + geom_point() + facet_wrap('student') + theme_light() + ggtitle("Conjunto de prueba") Ahora utilicemos la función tune junto con la función svm para seleccionar el mejor modelo de un conjunto de modelos, los modelos considerados serán aquellos obtenidos al variar los valores de los parámetros cost y gamma (usaremos un kernel radial). # Ahora utilizamos la función `tune` junto con la función `svm` para # seleccionar el mejor modelo de un conjunto de modelos, los modelos # considerados son aquellos obtenidos al variar los valores de los # parámetros `cost` y `gamma`. Kernel Radial

#tune.rad = tune(svm, default~., data = Default[train,], # kernel = "radial", # ranges = list( # cost = c(0.1, 1, 10, 100, 1000), # gamma = seq(0.01, 10, 0.5) # ) #)

## Se ha elegido el mejor modelo utilizando *validación cruzada de 10 # iteraciones*

## summary(tune.rad)

## Aquí un resumen del modelo seleccionado

## summary(tune.rad$best.model)

best <- svm(default~., data = Default[train,], kernel = "radial", cost = 100, gamma = 1.51 ) Con el mejor modelo seleccionado y utilizando el conjunto de prueba, obtengamos una matriz de confusión, para observar

el número de aciertos y errores cometidos por el modelo. También obtengamos la proporción total de aciertos y la matriz que muestre las proporciones de aciertos y errores cometidos pero por categorías. mc <- table(true = Default[-train, "default"], pred = predict(best, newdata = Default[-train,])) mc

## El porcentaje total de aciertos obtenido por el modelo usando el conjunto de prueba es el siguiente

round(sum(diag(mc))/sum(colSums(mc)), 5)

## Ahora observemos las siguientes proporciones

rs <- apply(mc, 1, sum) r1 <- round(mc[1,]/rs[1], 5) r2 <- round(mc[2,]/rs[2], 5) rbind(No=r1, Yes=r2) Ajustemos nuevamente el mejor modelo, pero ahora con el argumento decision.values = TRUE. Obtengamos los valores predichos para el conjunto de prueba utilizando el mejor modelo, las funciones predict, attributes y el argumento decision.values = TRUE dentro de predict. fit <- svm(default ~ ., data = Default[train,], kernel = "radial", cost = 100, gamma = 1.51, decision.values = TRUE)

fitted <- attributes(predict(fit, Default[-train,], decision.values = TRUE))$decision.values Realicemos clasificación de las observaciones del conjunto de prueba utilizando los valores predichos por el modelo y un umbral de decisión igual a cero. También obtengamos la matriz de confusión y proporciones como anteriormente hicimos. eti <- ifelse(fitted < 0, "Yes", "No")

mc <- table(true = Default[-train, "default"], pred = eti) mc

round(sum(diag(mc))/sum(colSums(mc)), 5)

rs <- apply(mc, 1, sum) r1 <- round(mc[1,]/rs[1], 5) r2 <- round(mc[2,]/rs[2], 5) rbind(No=r1, Yes=r2) Repitamos el paso 7 pero con un umbral de decisión diferente, de tal manera que se reduzca la proporción del error más grave para la compañía de tarjetas de crédito. eti <- ifelse(fitted < 1.002, "Yes", "No")

mc <- table(true = Default[-train, "default"], pred = eti) mc

round(sum(diag(mc))/sum(colSums(mc)), 5)

rs <- apply(mc, 1, sum) r1 <- round(mc[1,]/rs[1], 5) r2 <- round(mc[2,]/rs[2], 5) rbind(No=r1, Yes=r2)

```
# Ejemplo 3. Máquinas de vectores de soporte (Compañía de tarjetas de crédito)

# Paquetes de R utilizados

suppressMessages(suppressWarnings(library(dplyr)))
suppressMessages(suppressWarnings(library(e1071)))
suppressMessages(suppressWarnings(library(ggplot2)))
suppressMessages(suppressWarnings(library(ISLR)))

# 1. Observemos algunas características del data frame Default del paquete ISLR, con funciones tales co

?Default
```

```
starting httpd help server ... done
```

```r
head(Default)
```

```
  default student   balance    income
1      No      No  729.5265 44361.625
2      No     Yes  817.1804 12106.135
3      No      No 1073.5492 31767.139
4      No      No  529.2506 35704.494
5      No      No  785.6559 38463.496
6      No     Yes  919.5885  7491.559
```

```r
tail(Default)
```

```
      default student   balance   income
9995       No     Yes  172.4130 14955.94
9996       No      No  711.5550 52992.38
9997       No      No  757.9629 19660.72
9998       No      No  845.4120 58636.16
9999       No      No 1569.0091 36669.11
10000      No     Yes  200.9222 16862.95
```

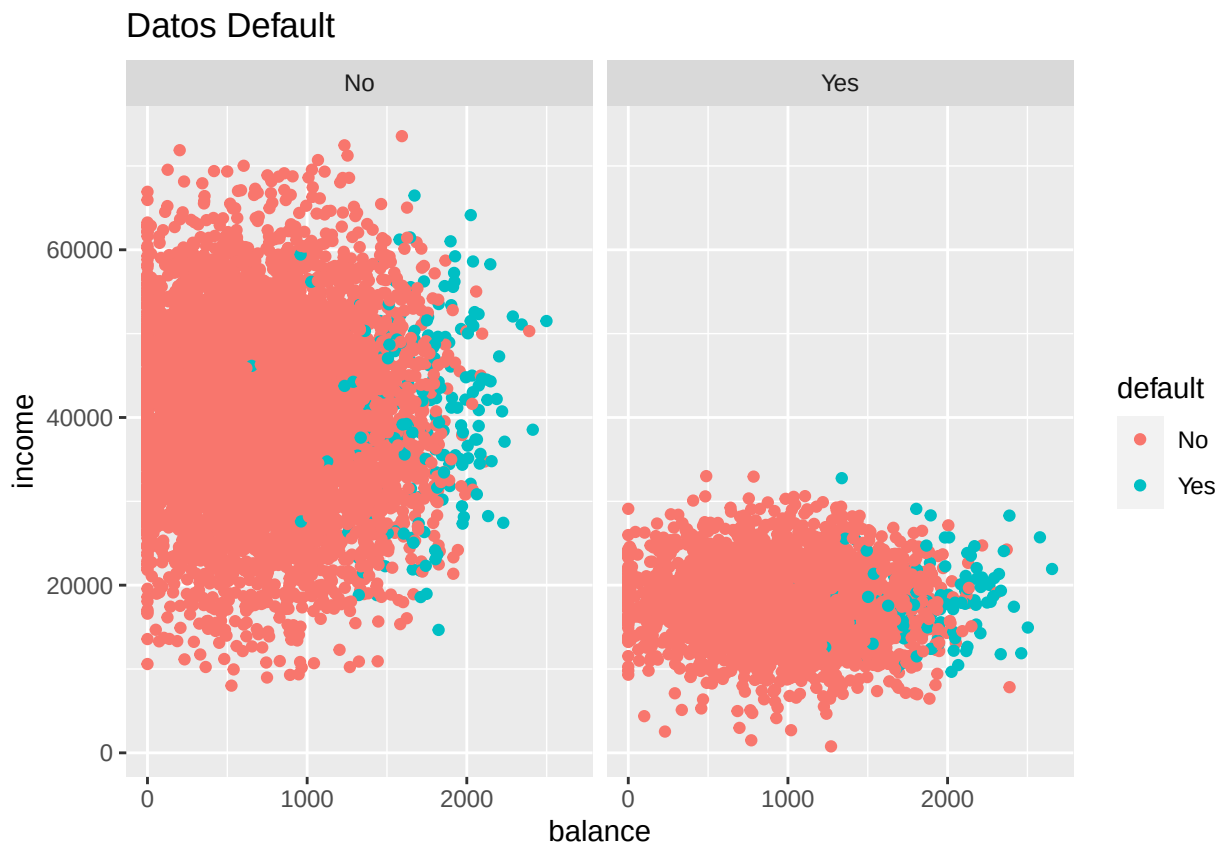
```r
dim(Default)
```

```
[1] 10000     4
```

```r
str(Default)
```

```
'data.frame':   10000 obs. of  4 variables:
 $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ student: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2 1 1 ...
 $ balance: num  730 817 1074 529 786 ...
 $ income : num  44362 12106 31767 35704 38463 ...
```

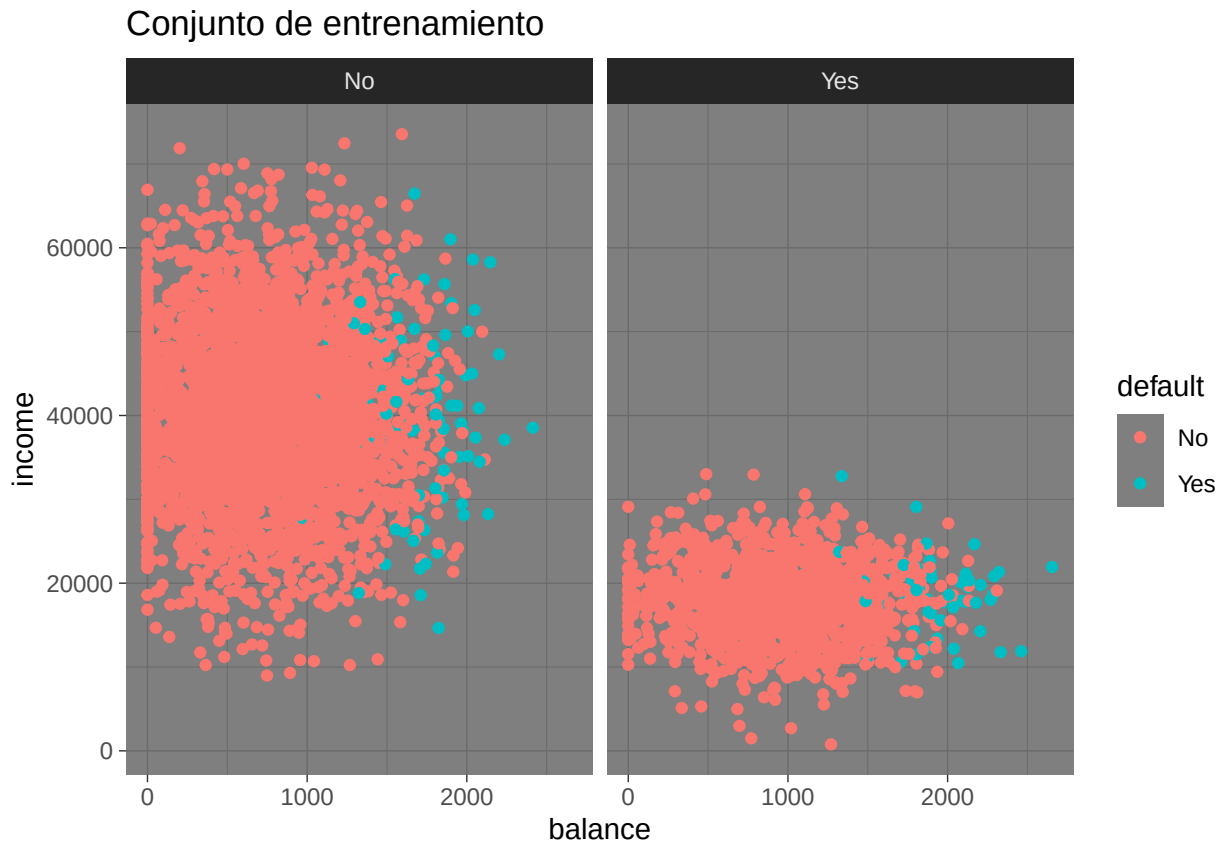
```r
# 2. Usando ggplot del paquete ggplot2, realicemos un gráfico de dispersión con la variable balance en

ggplot(Default, aes(x = balance, y = income, colour = default)) +
  geom_point() + facet_wrap('student') +
  theme_grey() + ggtitle("Datos Default")
```



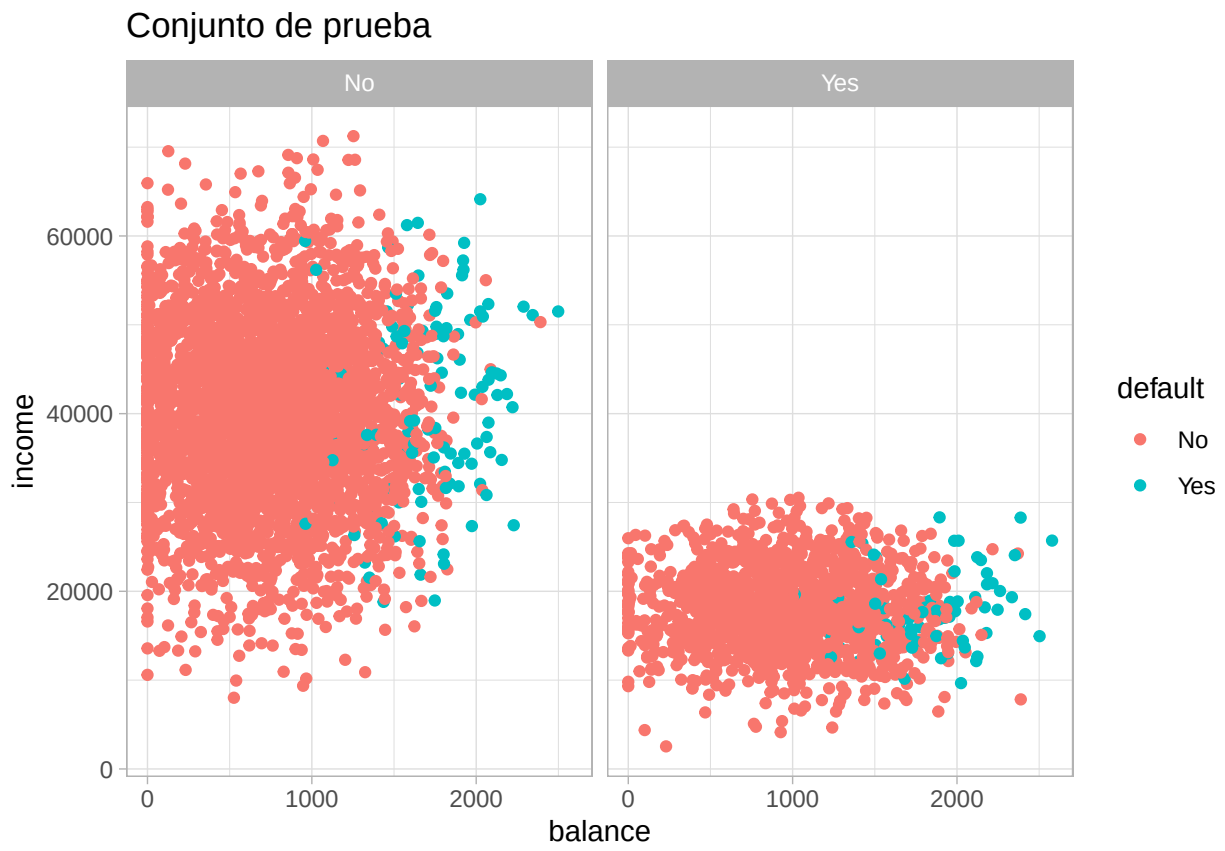

```
# 3. Generemos un vector de índices llamado train, tomando de manera aleatoria 5000 números de los prim

set.seed(2020)
train = sample(nrow(Default),
               round(nrow(Default)/2))
tail(Default[train, ])
```

```
     default student   balance   income
6258      No     Yes  733.7195 27165.48
7747      No      No  314.4592 40016.48
9268      No     Yes 1578.2896 12778.60
2481      No     Yes 1402.5539 16607.56
6527      No     Yes  635.3947 18344.08
5831      No      No  709.2575 23249.94
```

```
ggplot(Default[train, ],
       aes(x = balance, y = income, colour = default)) +
  geom_point() + facet_wrap('student') +
  theme_dark() + ggtitle("Conjunto de entrenamiento")
```




```
ggplot(Default[-train, ],
       aes(x = balance, y = income, colour = default)) +
  geom_point() + facet_wrap('student') +
  theme_light() + ggtitle("Conjunto de prueba")
```




```
# 4. Ahora utilicemos la función tune junto con la función svm para seleccionar el mejor modelo de un c

# Ahora utilizamos la función 'tune' junto con la función 'svm' para
# seleccionar el mejor modelo de un conjunto de modelos, los modelos
# considerados son aquellos obtenidos al variar los valores de los
# parámetros 'cost' y 'gamma'. Kernel Radial

#tune.rad = tune(svm, default~., data = Default[train,],
#                kernel = "radial",
#                ranges = list(
#                  cost = c(0.1, 1, 10, 100, 1000),
#                  gamma = seq(0.01, 10, 0.5)
#                )
#)

# Se ha elegido el mejor modelo utilizando *validación cruzada de 10
# iteraciones*

# summary(tune.rad)

# Aquí un resumen del modelo seleccionado

# summary(tune.rad$best.model)

best <- svm(default~.,  data = Default[train,],
            kernel = "radial",
```

```
            cost = 100,
            gamma = 1.51
            )

# 5. Con el mejor modelo seleccionado y utilizando el conjunto de prueba, obtengamos una matriz de conf

mc <- table(true = Default[-train, "default"],
            pred = predict(best,
                           newdata = Default[-train,]))
mc
```

```
     pred
true    No  Yes
  No  4803   17
  Yes  131   49
```

```
# El porcentaje total de aciertos obtenido por el modelo usando el
# conjunto de prueba es el siguiente

round(sum(diag(mc))/sum(colSums(mc)), 5)
```

```
[1] 0.9704
```

```
# Ahora observemos las siguientes proporciones

rs <- apply(mc, 1, sum)
r1 <- round(mc[1,]/rs[1], 5)
r2 <- round(mc[2,]/rs[2], 5)
rbind(No=r1, Yes=r2)
```

```
         No     Yes
No  0.99647 0.00353
Yes 0.72778 0.27222
```

```
# 6. Ajustemos nuevamente el mejor modelo, pero ahora con el argumento decision.values = TRUE. Obtengam

fit <- svm(default ~ ., data = Default[train,],
           kernel = "radial", cost = 100, gamma = 1.51,
           decision.values = TRUE)

fitted <- attributes(predict(fit, Default[-train,],
                             decision.values = TRUE))$decision.values

# 7. Realicemos clasificación de las observaciones del conjunto de prueba utilizando los valores predic

eti <- ifelse(fitted < 0, "Yes", "No")

mc <- table(true = Default[-train, "default"],
            pred = eti)
mc
```

```
     pred
true    No  Yes
  No  4803   17
  Yes  131   49
```

```
round(sum(diag(mc))/sum(colSums(mc)), 5)
```

```
[1] 0.9704
```

```
rs <- apply(mc, 1, sum)
r1 <- round(mc[1,]/rs[1], 5)
r2 <- round(mc[2,]/rs[2], 5)
rbind(No=r1, Yes=r2)
```

```
         No     Yes
No  0.99647 0.00353
Yes 0.72778 0.27222
```

```
# 8. Repitamos el paso 7 pero con un umbral de decisión diferente, de tal manera que se reduzca la prop

eti <- ifelse(fitted < 1.002, "Yes", "No")

mc <- table(true = Default[-train, "default"],
            pred = eti)
mc
```

```
     pred
true    No  Yes
  No  4163  657
  Yes   82   98
```

```
round(sum(diag(mc))/sum(colSums(mc)), 5)
```

```
[1] 0.8522
```

```
rs <- apply(mc, 1, sum)
r1 <- round(mc[1,]/rs[1], 5)
r2 <- round(mc[2,]/rs[2], 5)
rbind(No=r1, Yes=r2)
```

```
         No     Yes
No  0.86369 0.13631
Yes 0.45556 0.54444
```

# RETO 2 MAQUINAS DE VECTORES DE SOPORTE

## OBJETIVO

- Crear un conjunto de entrenamiento y uno de prueba a partir de un conjunto de datos dado
- Ajustar máquinas de vectores de soporte a un conjunto de entrenamiento
- Llevar a cabo clasificación con un conjunto de prueba y crear la matriz de confusión

## DESARROLLO

En el archivo de datos csv adjunto se encuentran observaciones correspondientes a dos clases diferentes indicadas por la variable y. Únicamente hay dos variables predictoras o características. A continuación realice los siguientes requerimientos (Hint: transforme primero la variable de respuesta y a variable categórica con las funciones mutate y factor):

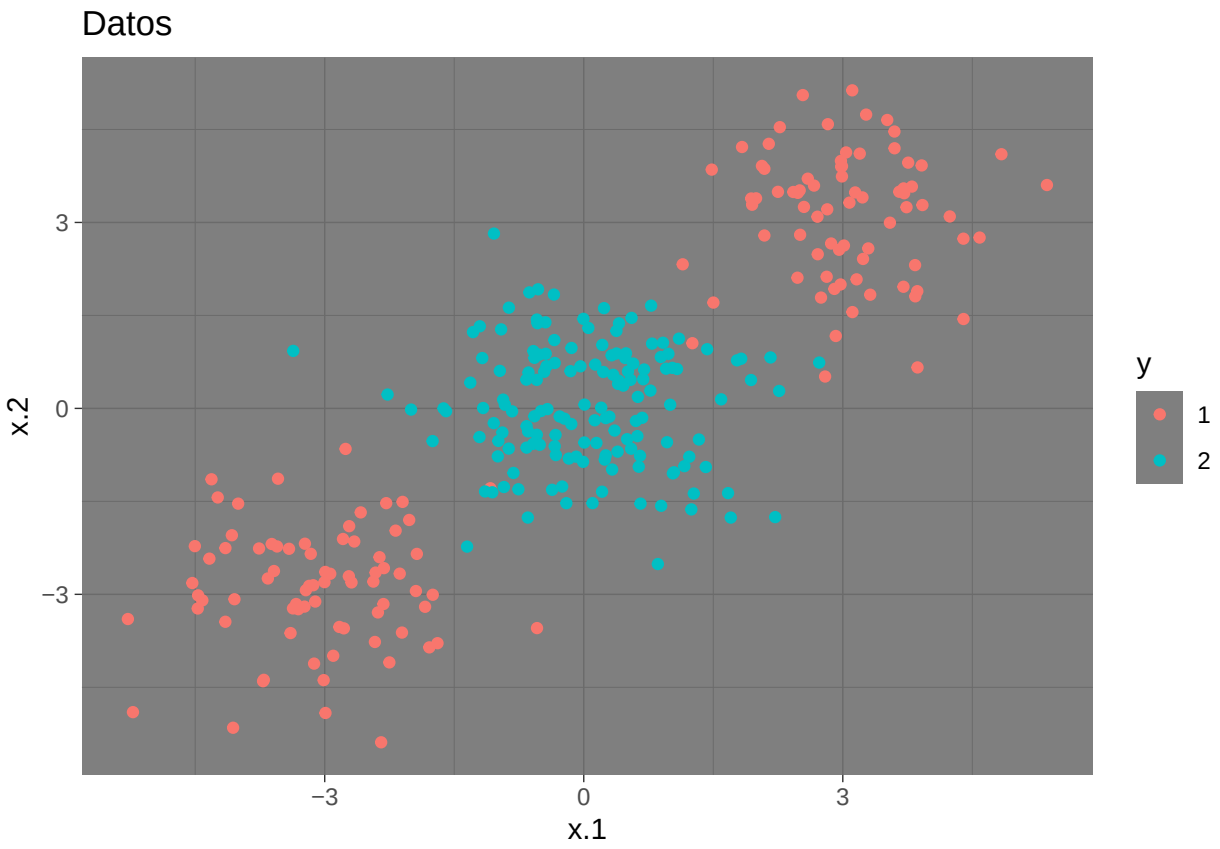
1. Carga los paquetes ggplot2 y e1071; observe algunas características del data frame con las funciones tail y dim. Obtenga el gráfico de dispersión de los datos diferenciando las dos clases.
2. Genera de manera aleatoria un vector de índices para filtrar un conjunto de entrenamiento a partir del conjunto de datos dado. Con ayuda de las funciones tune y svm ajuste máquinas de vectores de soporte con un kernel radial a los datos de entrenamiento, para valores del parámetro cost igual a 0.1, 1, 10, 100, 1000 y valores del parámetro gamma igual a 0.5, 1, 2, 3, 4. Obtenga un resumen de los resultados.
3. Con el modelo que tuvo el mejor desempeño en el paso anterior realiza clasificación con la función predict y el conjunto de datos de prueba. Muestre la matriz de confusión.

```
# Reto 2. Máquinas de vectores de soporte

# En el archivo de datos csv adjunto se encuentran observaciones correspondientes a dos clases diferent

# 1. Cargue los paquetes ggplot2 y e1071; observe algunas características
# del data frame con las funciones tail y dim. Obtenga el gráfico de
# dispersión de los datos diferenciando las dos clases.
# 2. Genere de manera aleatoria un vector de índices para filtrar un
# conjunto de entrenamiento a partir del conjunto de datos dado.
# Con ayuda de las funciones tune y svm ajuste máquinas de vectores
# de soporte con un kernel radial a los datos de entrenamiento,
# para valores del parámetro cost igual a 0.1, 1, 10, 100, 1000
# y valores del parámetro gamma igual a 0.5, 1, 2, 3, 4.
# Obtenga un resumen de los resultados.
# 3. Con el modelo que tuvo el mejor desempeño en el
# paso anterior realice clasificación con la función
# predict y el conjunto de datos de prueba. Muestre la matriz de confusión.

# **Solución**

# Primero establecemos nuestro directorio de trabajo en donde
# deberán estar nuestros datos.

datos <- read.csv("datosclases.csv")

# 1.

# Cargamos los paquetes `ggplot2` y `e1071`,
# observamos algunas características del data frame
# con las funciones `tail` y `dim`.

library(dplyr)
library(ggplot2)
library(e1071)
```

```
###

tail(datos); dim(datos); str(datos)
```

```
           x.1        x.2 y
295 -0.1534920  0.6005837 2
296  0.9802401  0.8803731 2
297  0.6513691 -0.7643731 2
298  1.0802927  0.6356774 2
299 -0.5358458  1.3744885 2
300  0.2345289  1.6176017 2

[1] 300   3

'data.frame':   300 obs. of  3 variables:
 $ x.1: num  3.32 2.83 2.92 2.48 2.54 ...
 $ x.2: num  1.83 4.59 1.17 3.48 5.06 ...
 $ y  : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
datos <- mutate(datos, y = factor(y))
# Obtenemos el gráfico de dispersión de los datos
# diferenciando las dos clases

ggplot(datos, aes(x = x.1, y = x.2, colour = y)) +
  geom_point() +
  theme_dark() + ggtitle("Datos")
```



```r
###

# 2.

# Generamos índices para el conjunto de entrenamiento

train <- sample(300, 150)
tail(as.data.frame(train))
```

```
    train
145   112
146    65
147   238
148   235
149    69
150   117
```

```r
###

# Ajustamos máquinas de vectores de soporte con un kernel radial
# para diferentes valores de los parámetros 'cost' y 'gamma'

set.seed(67)
tune.out <- tune(svm, y~., data = datos[train, ],
                 kernel = "radial",
                 ranges = list(cost = c(0.1, 1, 10, 100, 1000),
                               gamma = c(0.5, 1, 2, 3, 4)))


### Obtenemos un resumen de los modelos ajustados y su desempeño

summary(tune.out)
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
   10     3

- best performance: 0

- Detailed performance results:
    cost gamma       error dispersion
1  1e-01   0.5 0.020000000 0.04499657
2  1e+00   0.5 0.006666667 0.02108185
3  1e+01   0.5 0.006666667 0.02108185
4  1e+02   0.5 0.013333333 0.04216370
5  1e+03   0.5 0.013333333 0.04216370
6  1e-01   1.0 0.013333333 0.02810913
7  1e+00   1.0 0.020000000 0.04499657
```

```
8  1e+01   1.0 0.013333333 0.04216370
9  1e+02   1.0 0.013333333 0.04216370
10 1e+03   1.0 0.013333333 0.04216370
11 1e-01   2.0 0.013333333 0.02810913
12 1e+00   2.0 0.006666667 0.02108185
13 1e+01   2.0 0.006666667 0.02108185
14 1e+02   2.0 0.006666667 0.02108185
15 1e+03   2.0 0.006666667 0.02108185
16 1e-01   3.0 0.020000000 0.04499657
17 1e+00   3.0 0.006666667 0.02108185
18 1e+01   3.0 0.000000000 0.00000000
19 1e+02   3.0 0.000000000 0.00000000
20 1e+03   3.0 0.000000000 0.00000000
21 1e-01   4.0 0.020000000 0.04499657
22 1e+00   4.0 0.006666667 0.02108185
23 1e+01   4.0 0.000000000 0.00000000
24 1e+02   4.0 0.000000000 0.00000000
25 1e+03   4.0 0.000000000 0.00000000
```

```r
###

# Realizamos clasificación con el mejor modelo ajustado y obtenemos
# la matriz de confusión.

table(true = datos[-train, "y"],
      pred = predict(tune.out$best.model, newdata = datos[-train,]))
```

```
    pred
true  1  2
   1 73  2
   2  4 71
```

# EJEMPLO 4. MAQUINAS DE VECTORES DE SOPORTE

## OBJETIVO

- Conocer algunas funciones de R que nos ayudarán a llevar a cabo clasificaciones. Aprenderemos a dividir un conjunto de datos dado, en dos conjuntos, uno llamado el conjunto de entrenamiento y el otro llamado el conjunto de prueba; desarrollaremos un clasificador con ayuda de R y del conjunto de entrenamiento y evaluaremos su desempeño con el conjunto de prueba. En la práctica, un clasificador de esta naturaleza podría ser usado para ayudar a hacer diagnósticos de enfermedades, para decidir a quien otorgarle un crédito o a quien no y en general para clasificar personas en una de dos o más categorías.

## DESARROLLO

Clasificador de vectores de soporte Vamos a comenzar cargando el paquete e1071 para ajustar máquinas de vectores de soporte

## install.packages("e1071") para instalarlo

library(e1071) Generamos observaciones correspondientes a dos clases

set.seed(754) x <- matrix(rnorm(30*2), ncol = 2) y <- c(rep(-1, 15), rep(1, 15)) x[y == 1, ] <- x[y == 1, ] + 1 plot(x, col = (3-y), pch = 16) Creamos un data frame con la respuesta como factor, está nos ayudará a realizar la clasificación

dat <- data.frame(x = x, y = as.factor(y)) tail(dat) Ajustamos el clasificador de vectores de soporte con la función svm

svmfit <- svm(y~., data = dat, kernel = "linear", cost = 10, scale = FALSE) Acontinuación, mostramos el clasificador de vectores de soporte junto con las observaciones. Los vectores de soporte se muestran como x's

plot(svmfit, dat) También podemos observar los índices (números de filas en el data frame) que corresponden a vectores de soporte

svmfit$index length(svmfit$index) Mostramos un breve resumen del ajuste

summary(svmfit) Volvemos a realizar el ajuste pero ahora con el valor del parámetro cost = 0.1

svmfit <- svm(y~., data = dat, kernel = "linear", cost = 0.1, scale = FALSE) Se grafica el clasificador

plot(svmfit, dat) Tenemos más vectores de soporte

length(svmfit$index) svmfit$index El siguiente comando indica que queremos comparar MVS con un kernel lineal, usando un rango de valores del parámetro cost

set.seed(524) tune.out <- tune(svm, y~., data = dat, kernel = "linear", ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100))) summary(tune.out) Elegimos el mejor modelo ajustado

bestmod <- tune.out$best.model summary(bestmod) Ahora consideramos un conjunto de datos de prueba para poder evaluar nuestro clasificador

xtest <- matrix(rnorm(45*2), ncol = 2) ytest <- sample(c(-1, 1), 45, rep = TRUE) xtest[ytest == 1, ] <- xtest[ytest == 1, ] + 1 testdat <- data.frame(x = xtest, y = as.factor(ytest)) tail(testdat) Realizamos una clasificación usando el mejor modelo ajustado y el conjunto de datos de prueba. Luego, mostramos la matriz de confusión

ypred <- predict(bestmod, testdat) table(predict = ypred, truth = testdat$y) Máquinas de vectores de soporte Generamos datos con una frontera de clase no lineal

set.seed(6891) x <- matrix(rnorm(200*2), ncol = 2) x[1:100,] <- x[1:100,] + 2 x[101:150,] <- x[101:150,] - 2 y <- c(rep(1, 150), rep(2, 50)) dat <- data.frame(x = x, y = as.factor(y)) head(dat) plot(x, col = y, pch = 16) Generamos índices para el conjunto de entrenamiento

train <- sample(200, 100) tail(as.data.frame(train)) Ajustamos una máquina de vectores de soporte con un kernel radial y valores de los parámetros gamma = 1 y cost = 1

svmfit <- svm(y~., data = dat[train, ], kernel = "radial", gamma = 1, cost = 1)

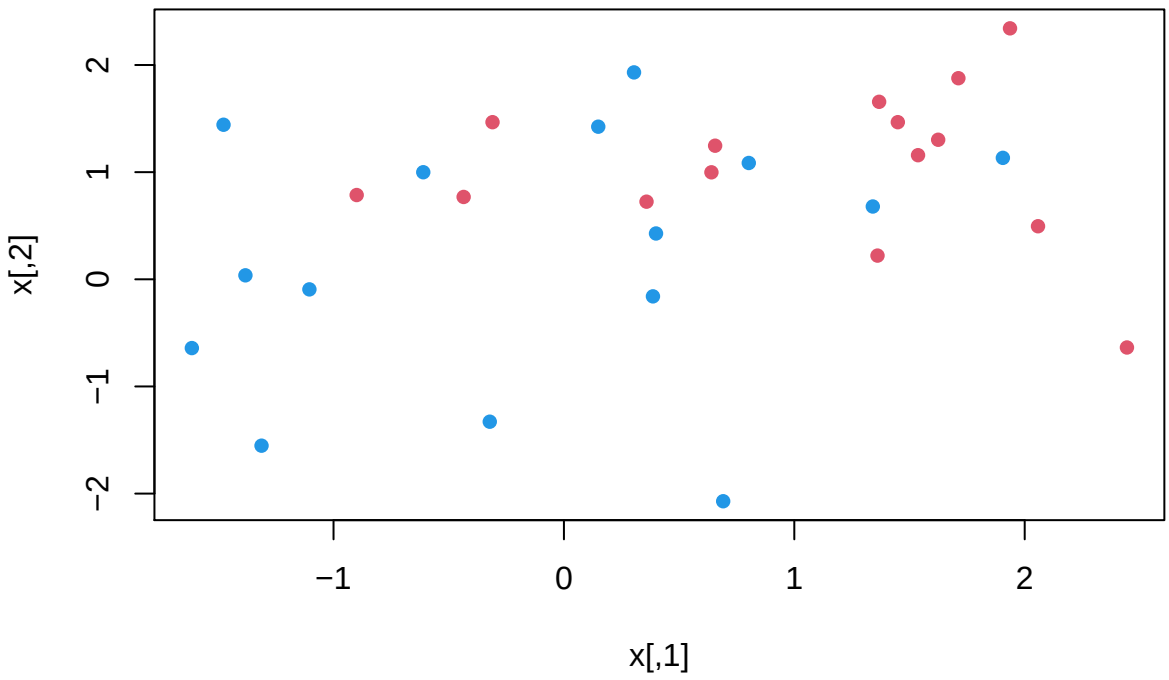
plot(svmfit, dat[train, ]) summary(svmfit) Ajustamos una máquina de vectores de soporte con un kernel radial y valores de los parámetros gamma = 1 y cost = 1e5

svmfit <- svm(y~., data = dat[train, ], kernel = "radial", gamma = 1, cost = 1e5) plot(svmfit, dat[train, ]) Ajustamos máquinas de vectores de soporte con un kernel radial para diferentes valores de los parámetros cost y gamma

set.seed(1980) tune.out <- tune(svm, y~., data = dat[train, ], kernel = "radial", ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5, 1, 2, 3, 4))) summary(tune.out) Realizamos clasificación con el mejor modelo ajustado y obtenemos la matriz de confusión, esta matriz servirá para conocer los valores ajustados correctamente

table(true = dat[-train, "y"], pred = predict(tune.out$best.model, newdata = dat[-train,]))

```r
# Ejemplo 4. Máquinas de Vectores de Soporte

#### Clasificador de vectores de soporte

# Cargamos el paquete 'e1071' para ajustar máquinas de vectores de soporte


# install.packages("e1071") para instalarlo
library(e1071)


###

# Generamos observaciones correspondientes a dos clases


set.seed(754)
x <- matrix(rnorm(30*2), ncol = 2)
y <- c(rep(-1, 15), rep(1, 15))
x[y == 1, ] <- x[y == 1, ] + 1


###


plot(x, col = (3-y), pch = 16)
```

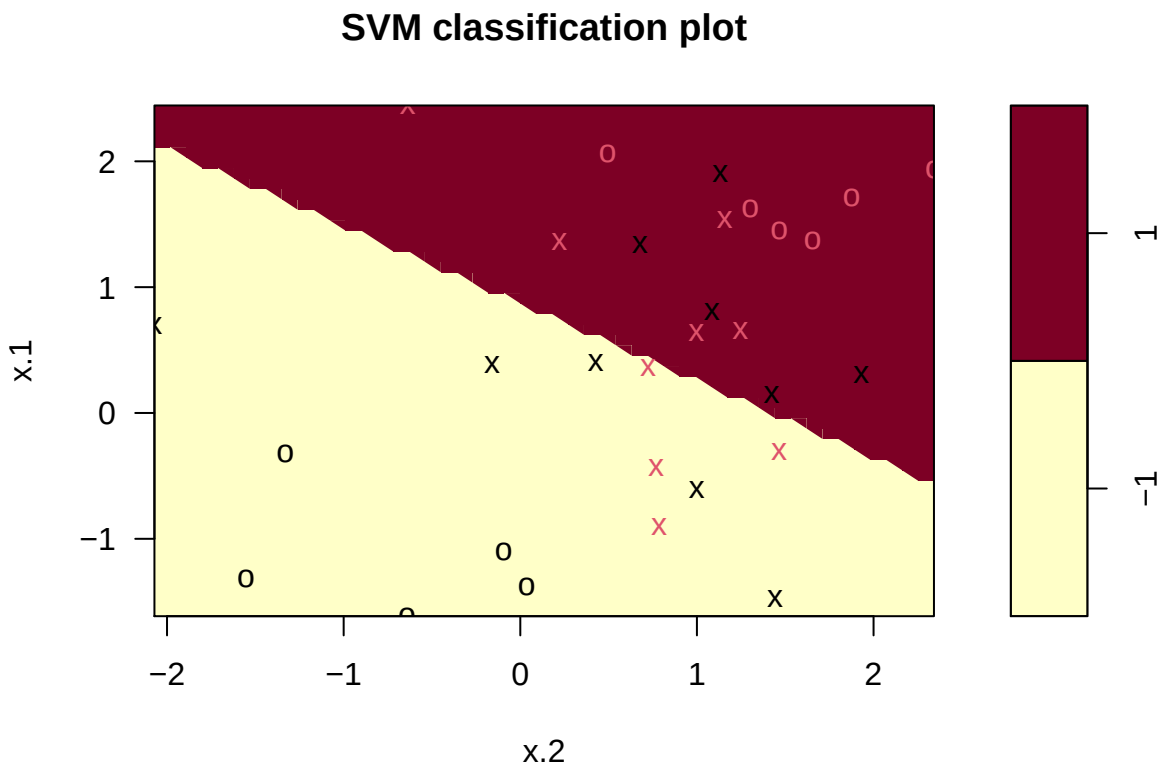



```
###

# Creamos un data frame con la respuesta como factor


dat <- data.frame(x = x, y = as.factor(y))
tail(dat)
```

```
         x.1       x.2 y
25 2.0577155 0.4951834 1
26 0.6402795 0.9984085 1
27 0.6562678 1.2468602 1
28 1.5369688 1.1582702 1
29 1.4492490 1.4664123 1
30 0.3584979 0.7244615 1
```

```
###

# Ajustamos el clasificador de vectores de soporte con la función 'svm'


svmfit <- svm(y~., data = dat, kernel = "linear",
              cost = 10, scale = FALSE)
```

```
# Acontinuación, mostramos el clasificador de vectores de soporte junto con las observaciones. Los vect

###

plot(svmfit, dat)
```




```
###

# También podemos observar los índices (números de filas en el data frame) que corresponden a vectores

svmfit$index
```

```
 [1]  2  5  8  9 10 11 12 13 14 15 17 20 21 23 24 26 27 28 30
```

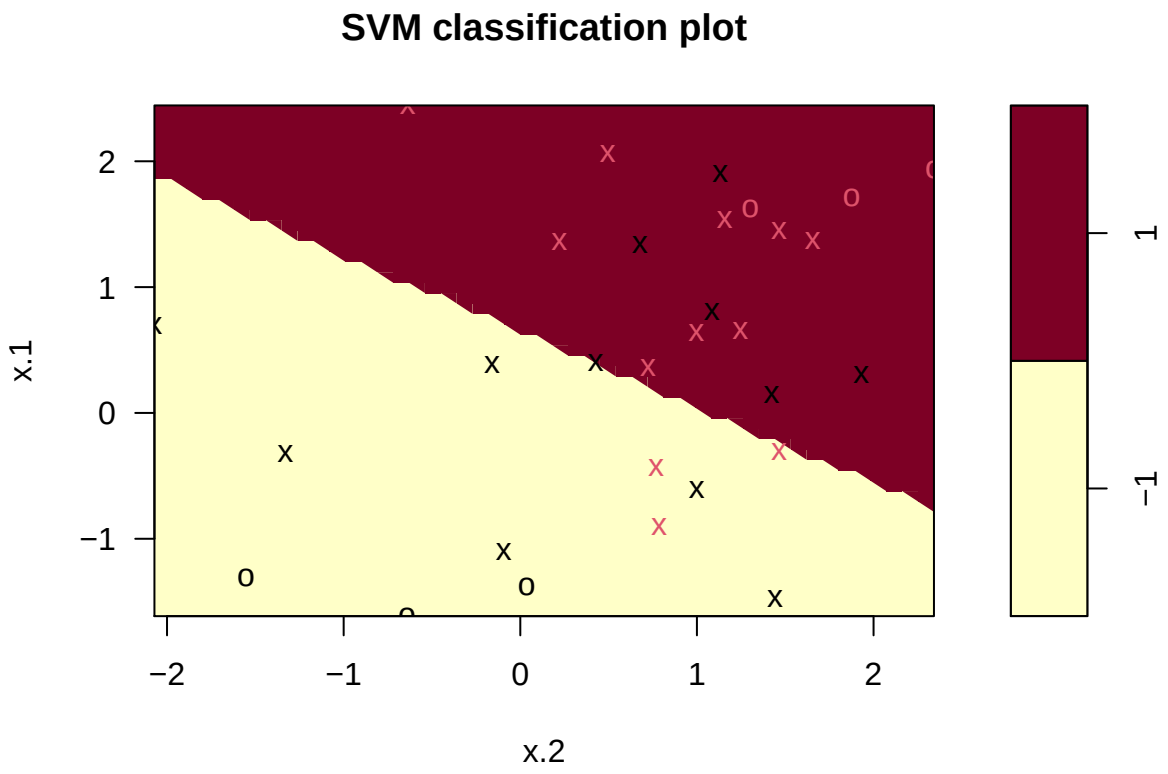
```
length(svmfit$index)
```

```
[1] 19
```

```
###

# Mostramos un breve resumen del ajuste

summary(svmfit)
```

```
Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10

Number of Support Vectors:  19

 ( 10 9 )


Number of Classes:  2

Levels:
 -1 1
```

```r
###

# Volvemos a realizar el ajuste pero ahora con el valor del parámetro `cost = 0.1`


svmfit <- svm(y~., data = dat, kernel = "linear",
cost = 0.1, scale = FALSE)


###


plot(svmfit, dat)
```




```
###

# Tenemos más vectores de soporte


length(svmfit$index)
```

```
[1] 24
```

```
svmfit$index
```

```
 [1]  2  4  5  7  8  9 10 11 12 13 14 15 17 20 21 22 23 24 25 26 27 28 29 30
```

```
###

# El siguiente comando indica que queremos comparar MVS con un kernel lineal, usando un rago de valores

set.seed(524)
tune.out <- tune(svm, y~., data = dat, kernel = "linear",
ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))


###

summary(tune.out)
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.3

- Detailed performance results:
   cost     error dispersion
1 1e-03 0.7333333  0.1405457
2 1e-02 0.7333333  0.1405457
3 1e-01 0.3000000  0.2459549
4 1e+00 0.3000000  0.2459549
5 5e+00 0.3000000  0.2459549
6 1e+01 0.3000000  0.2459549
7 1e+02 0.3000000  0.2459549
```

```r
###

# Elegimos el mejor modelo ajustado


bestmod <- tune.out$best.model


###


summary(bestmod)
```

```
Call:
best.tune(method = svm, train.x = y ~ ., data = dat, ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1

Number of Support Vectors:  24

 ( 12 12 )


Number of Classes:  2

Levels:
 -1 1
```

```
###

# Ahora consideramos un conjunto de datos de prueba


xtest <- matrix(rnorm(45*2), ncol = 2)
ytest <- sample(c(-1, 1), 45, rep = TRUE)
xtest[ytest == 1, ] <- xtest[ytest == 1, ] + 1
testdat <- data.frame(x = xtest, y = as.factor(ytest))
tail(testdat)
```

```
          x.1         x.2  y
40 -0.2862996 -0.09202137  1
41 -0.5867107 -1.58236689 -1
42  0.4137545  0.81011299  1
43  0.5243919  1.17268193  1
44 -1.2292391  0.13220739 -1
45  2.0949601  2.17074880  1
```

```
###

# Realizamos una clasificación usando el mejor modelo ajustado y el conjunto de datos de prueba. Luego,


ypred <- predict(bestmod, testdat)
table(predict = ypred, truth = testdat$y)
```

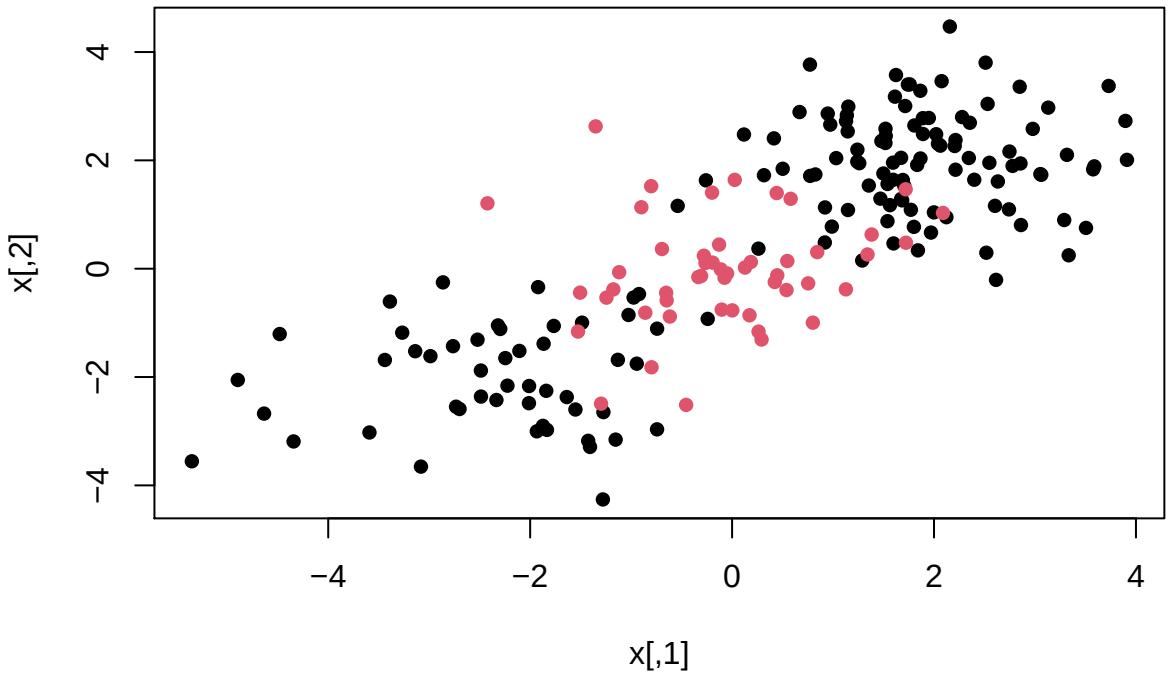
```
       truth
predict -1  1
     -1 11  5
     1   9 20
```

```
###

#### Máquinas de vectores de soporte

# Generamos datos con una frontera de clase no lineal


set.seed(6891)
x <- matrix(rnorm(200*2), ncol = 2)
x[1:100,] <- x[1:100,] + 2
x[101:150,] <- x[101:150,] - 2
y <- c(rep(1, 150), rep(2, 50))
dat <- data.frame(x = x, y = as.factor(y))
head(dat)
```

```
        x.1       x.2 y
1 1.8004687 0.7723837 1
2 0.7710962 3.7682197 1
3 1.5637161 1.1736177 1
4 2.6321386 1.6101621 1
```

```
5 2.5481400 1.9553106 1
6 1.1451845 2.5345495 1
```

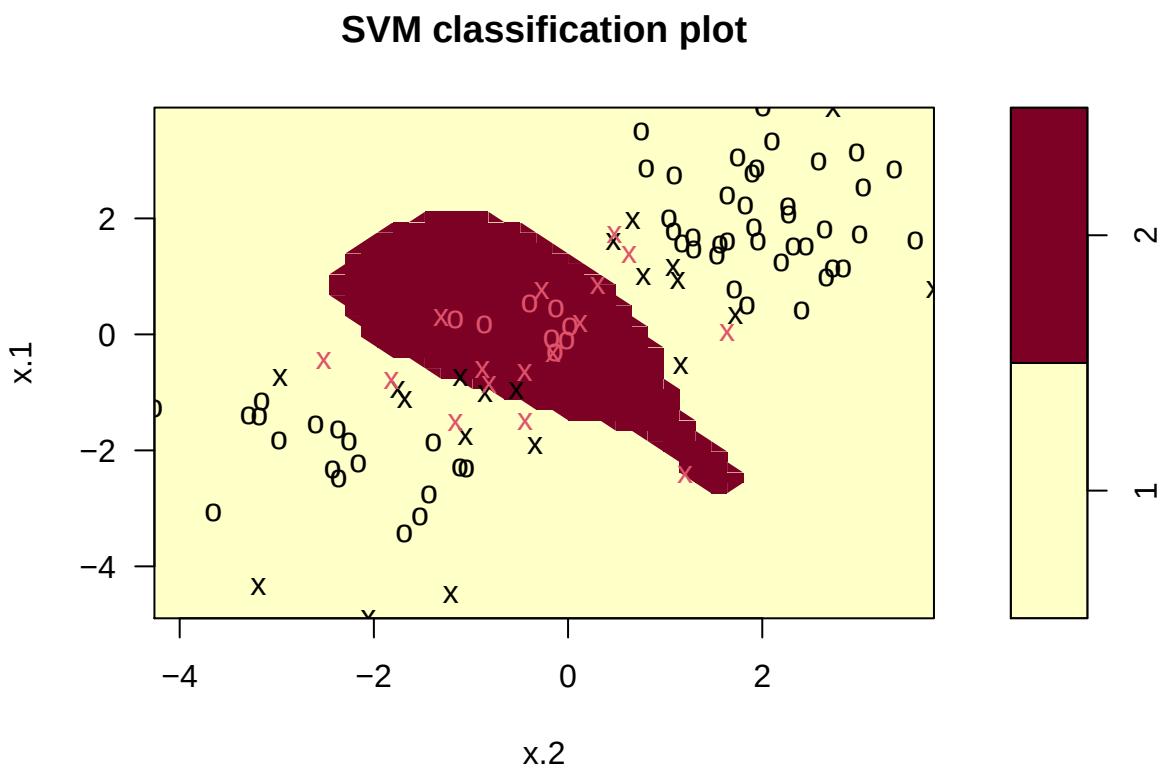
```
###


plot(x, col = y, pch = 16)
```




```
###

# Generamos índices para el conjunto de entrenamiento

train <- sample(200, 100)
tail(as.data.frame(train))
```

```
    train
95    197
96     43
97    167
98    131
99     17
100   172
```

```
###

# Ajustamos una máquina de vectores de soporte con un kernel radial y valores de los parámetros 'gamma
svmfit <- svm(y~., data = dat[train, ],
kernel = "radial", gamma = 1, cost = 1)


###


plot(svmfit, dat[train, ])
```

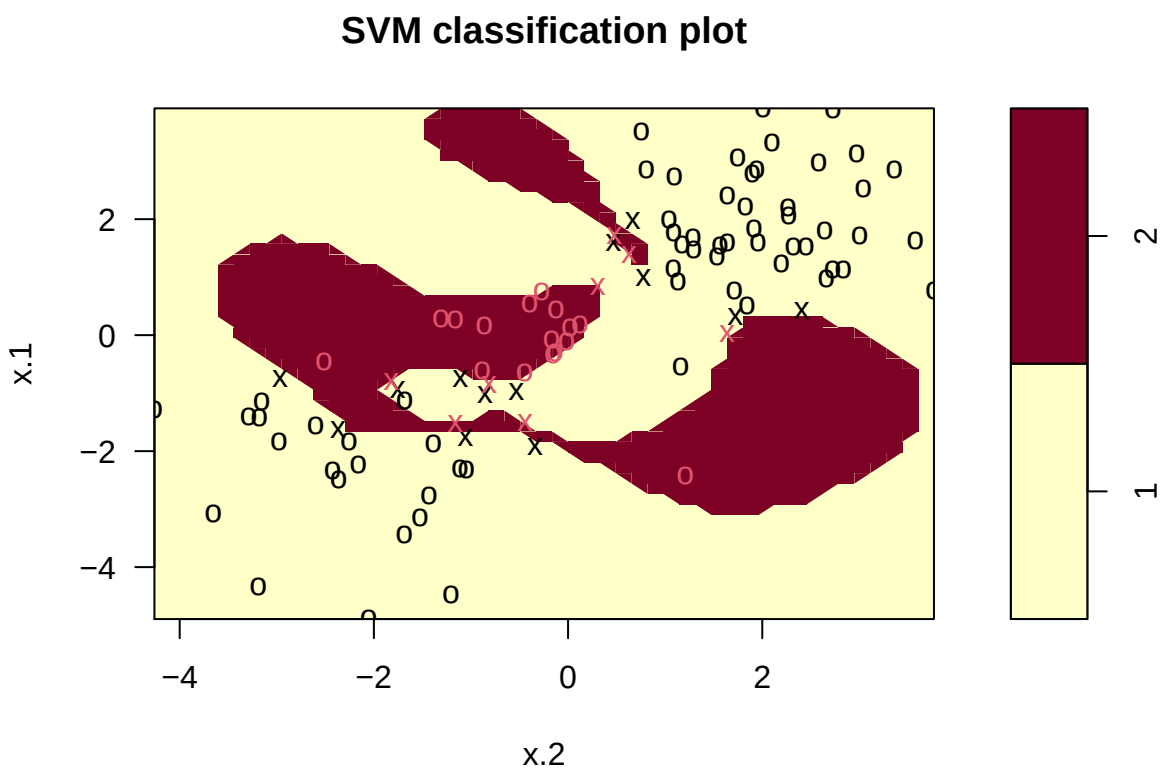



```
###

summary(svmfit)
```

```
Call:
svm(formula = y ~ ., data = dat[train, ], kernel = "radial", gamma = 1,
    cost = 1)


Parameters:
   SVM-Type:  C-classification
```

```
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  36

 ( 20 16 )


Number of Classes:  2

Levels:
 1 2
```

```
###

# Ajustamos una máquina de vectores de soporte con un kernel radial y valores de los parámetros 'gamma
svmfit <- svm(y~., data = dat[train, ],
kernel = "radial", gamma = 1, cost = 1e5)


###


plot(svmfit, dat[train, ])
```



```r
###

# Ajustamos máquinas de vectores de soporte con un kernel radial para diferentes valores de los parámet

set.seed(1980)
tune.out <- tune(svm, y~., data = dat[train, ], kernel = "radial",
ranges = list(cost = c(0.1, 1, 10, 100, 1000),
gamma = c(0.5, 1, 2, 3, 4)))


###

summary(tune.out)
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1   0.5

- best performance: 0.12

- Detailed performance results:
    cost gamma error dispersion
1  1e-01   0.5  0.24  0.1173788
2  1e+00   0.5  0.12  0.1813529
3  1e+01   0.5  0.14  0.1776388
4  1e+02   0.5  0.17  0.1888562
5  1e+03   0.5  0.18  0.1932184
6  1e-01   1.0  0.24  0.1173788
7  1e+00   1.0  0.12  0.1813529
8  1e+01   1.0  0.16  0.1712698
9  1e+02   1.0  0.18  0.1873796
10 1e+03   1.0  0.20  0.1763834
11 1e-01   2.0  0.24  0.1173788
12 1e+00   2.0  0.12  0.1549193
13 1e+01   2.0  0.18  0.1549193
14 1e+02   2.0  0.20  0.1763834
15 1e+03   2.0  0.19  0.1791957
16 1e-01   3.0  0.24  0.1173788
17 1e+00   3.0  0.12  0.1813529
18 1e+01   3.0  0.19  0.1791957
19 1e+02   3.0  0.20  0.1763834
20 1e+03   3.0  0.21  0.1969207
21 1e-01   4.0  0.24  0.1173788
22 1e+00   4.0  0.13  0.1766981
23 1e+01   4.0  0.18  0.1813529
24 1e+02   4.0  0.20  0.1699673
25 1e+03   4.0  0.22  0.1873796
```

```
###

# Realizamos clasificación con el mejor modelo ajustado y obtenemos la matriz de confusión.

table(true = dat[-train, "y"],
pred = predict(tune.out$best.model, newdata = dat[-train,]))
```

```
    pred
true  1  2
   1 69  5
   2  8 18
```