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

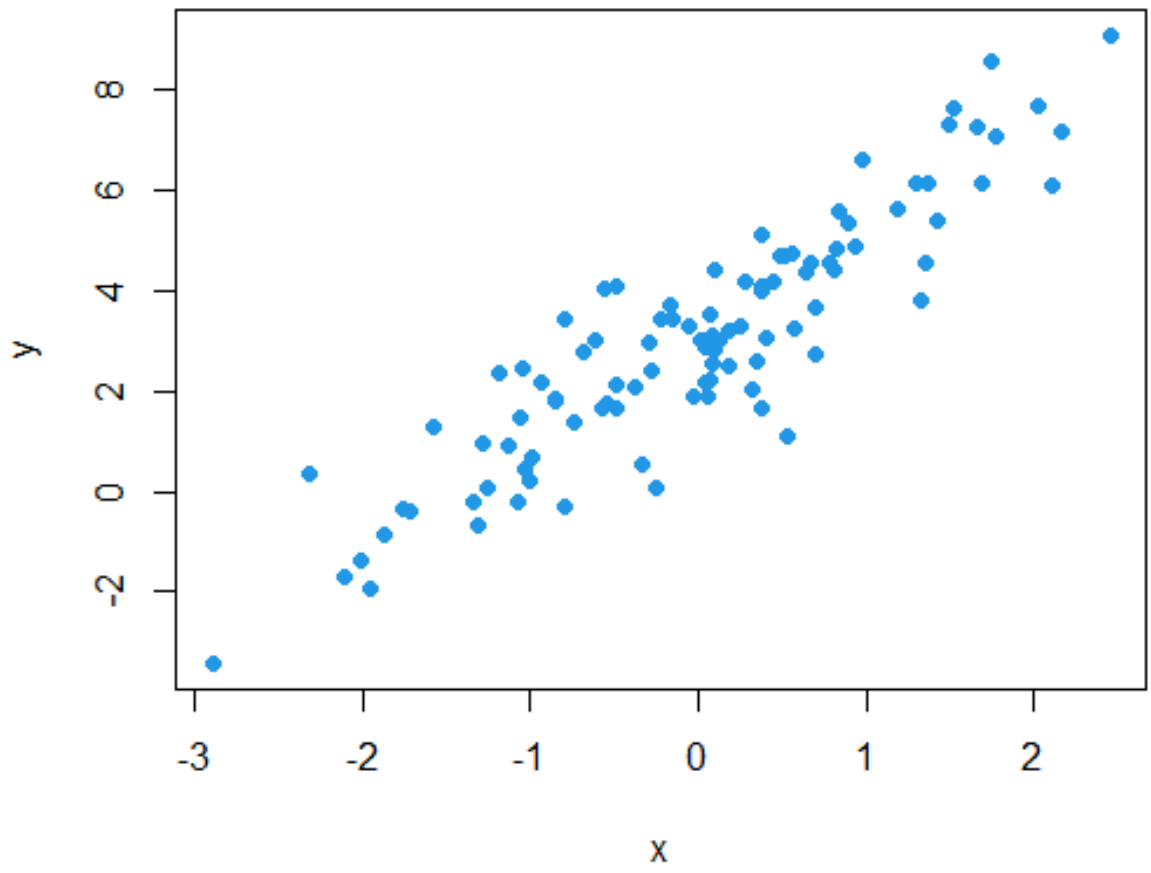


Figure 2: Ejemplo de regresion lineal

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

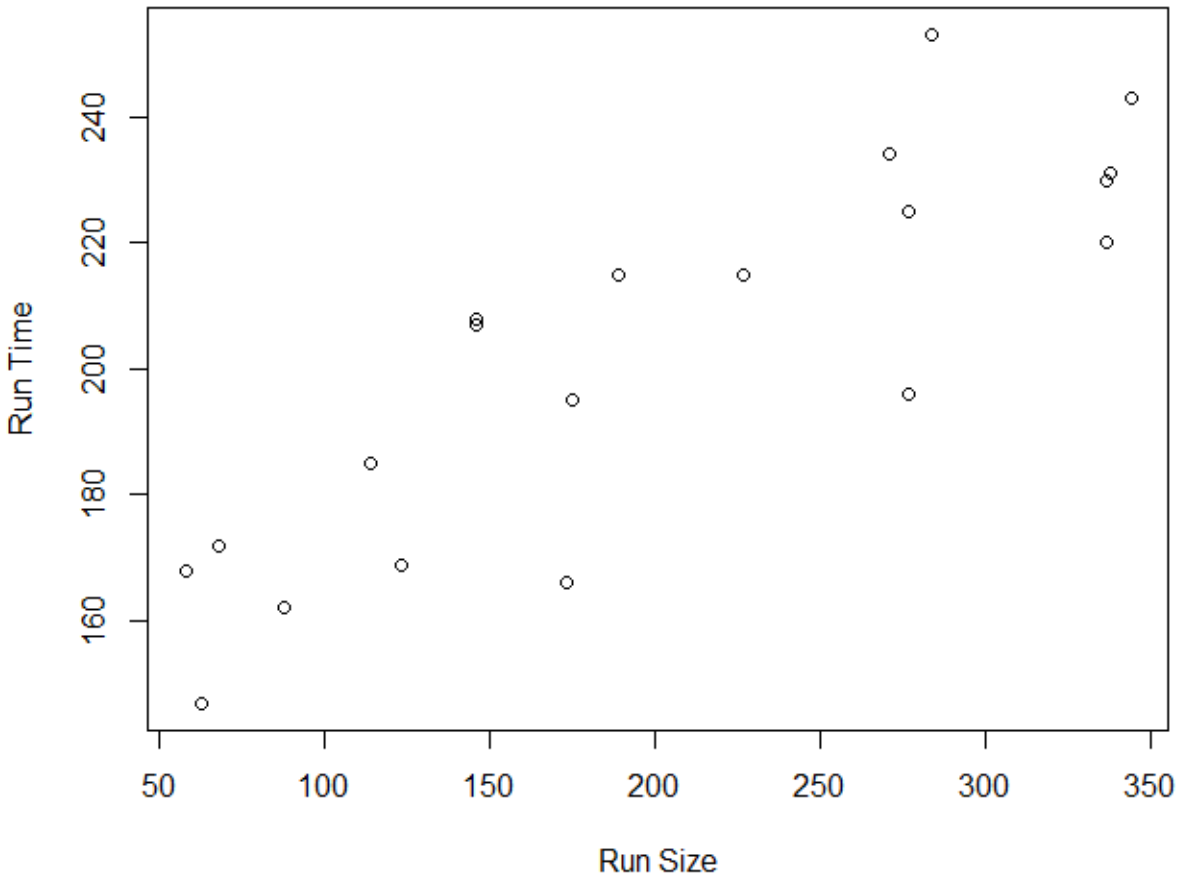


Figure 4: grafica

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

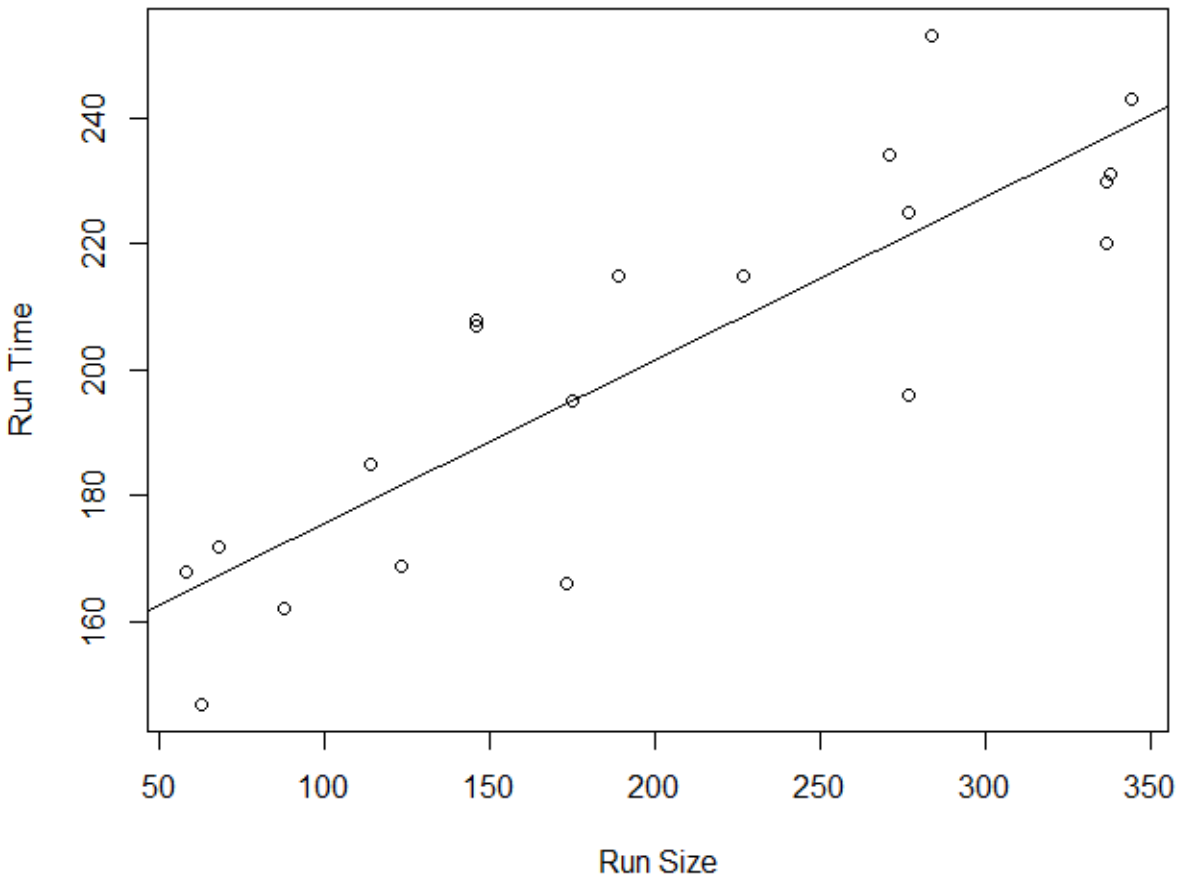


Figure 7: Imagen3

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

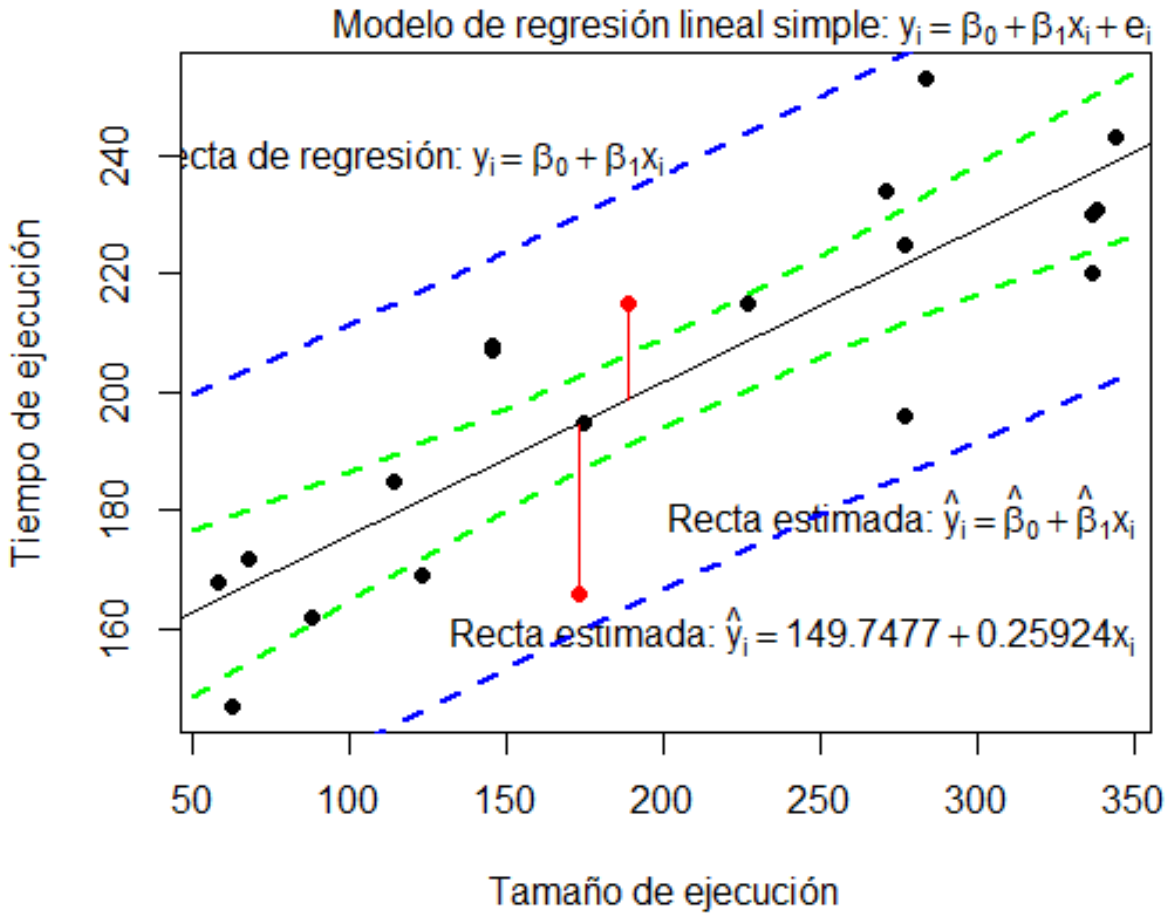


Figure 9: Imagen5

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

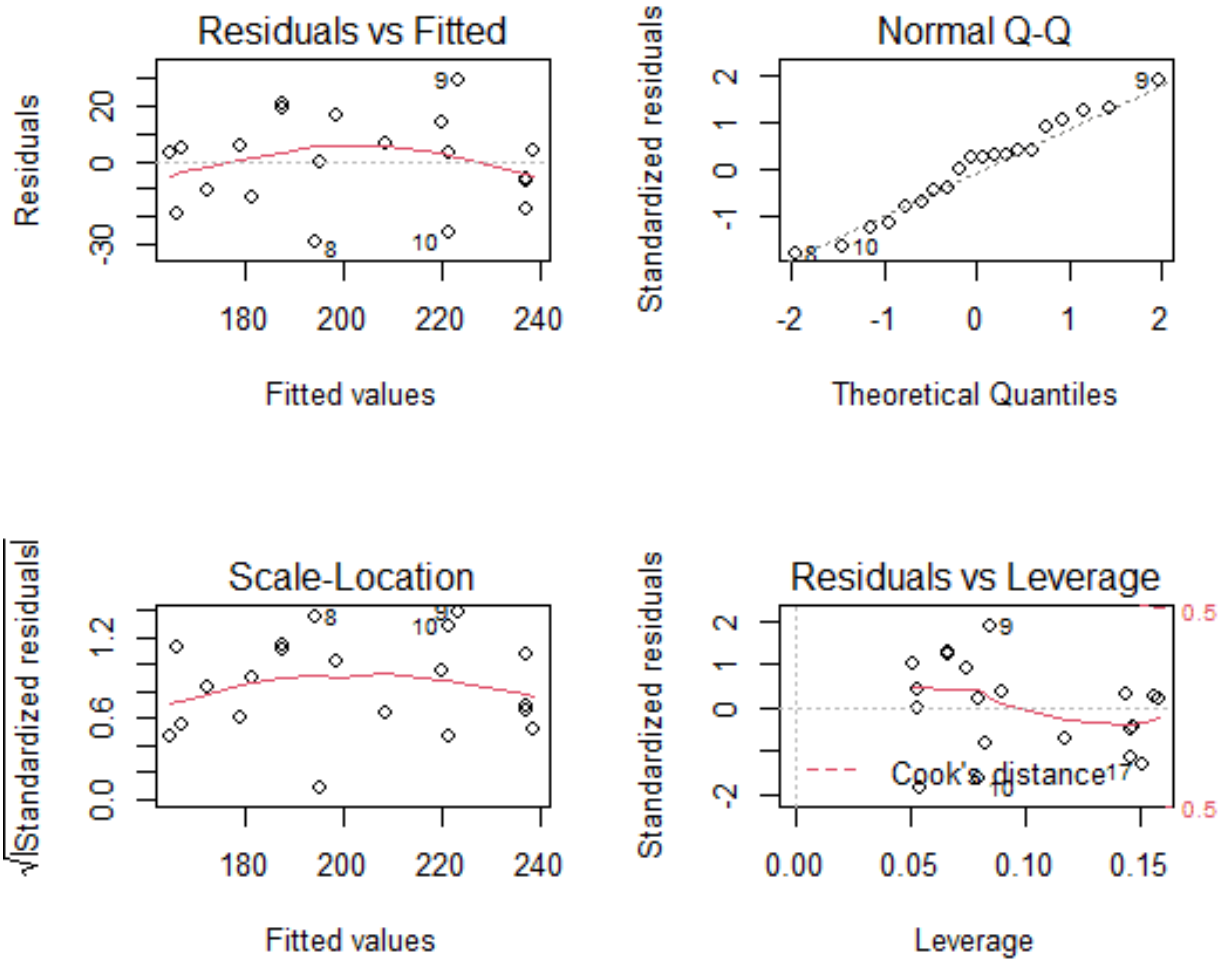


Figure 11: Imagen7

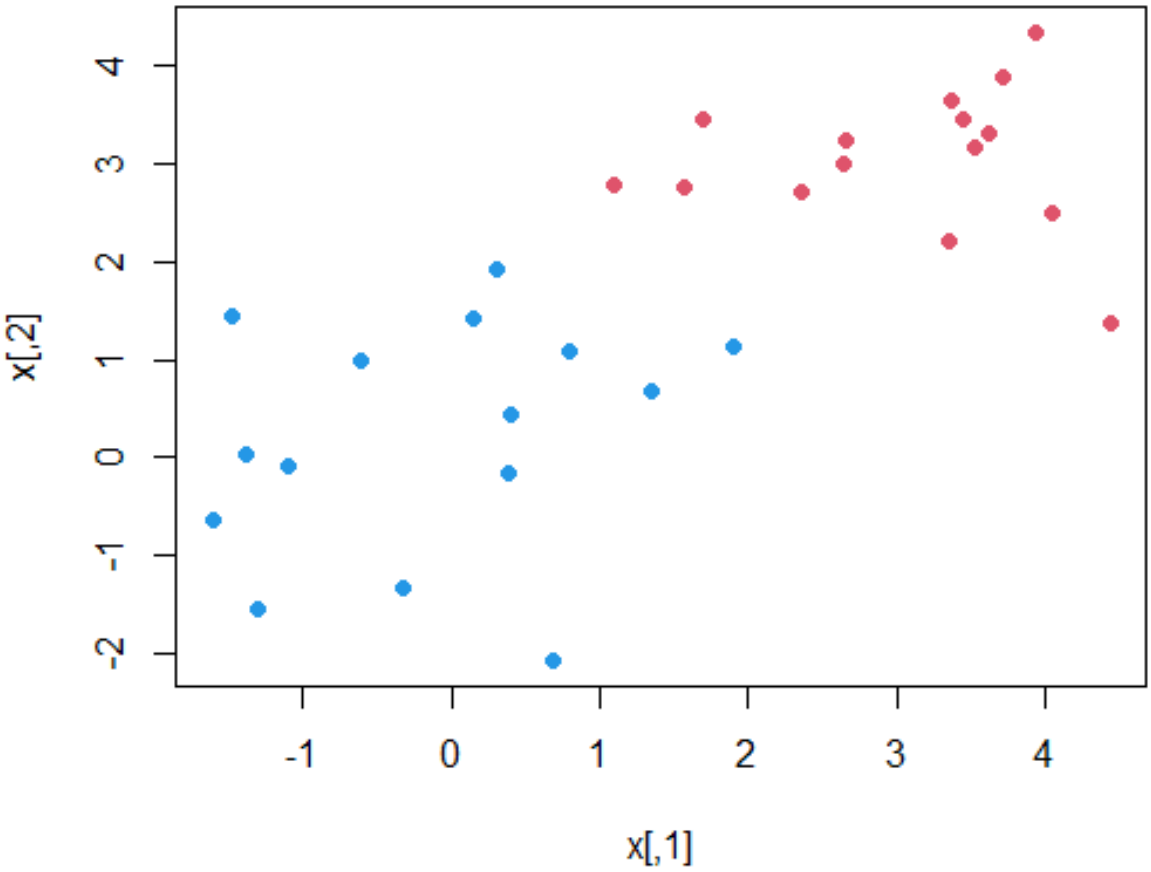


Figure 12: Imagen8

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

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.

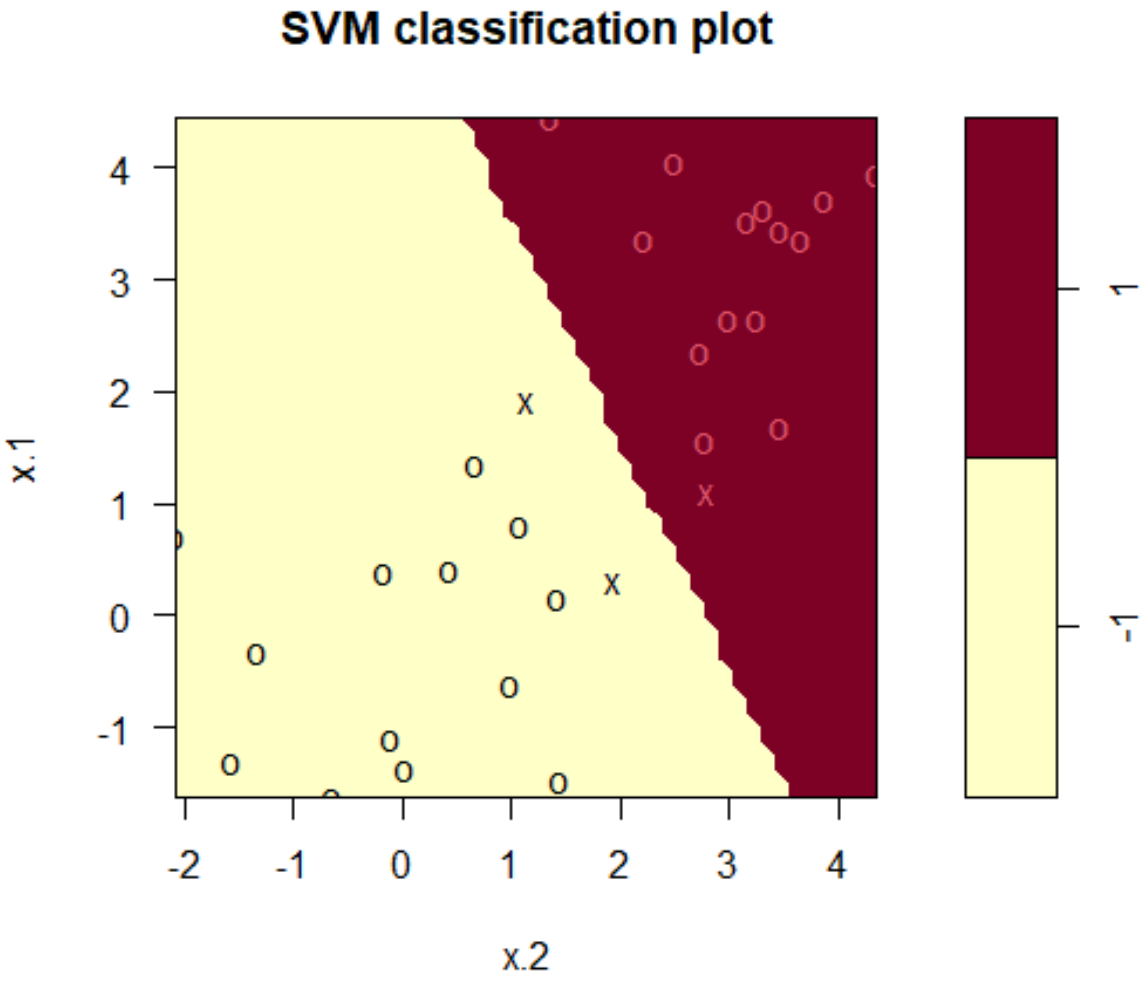


Figure 22: Imagen18

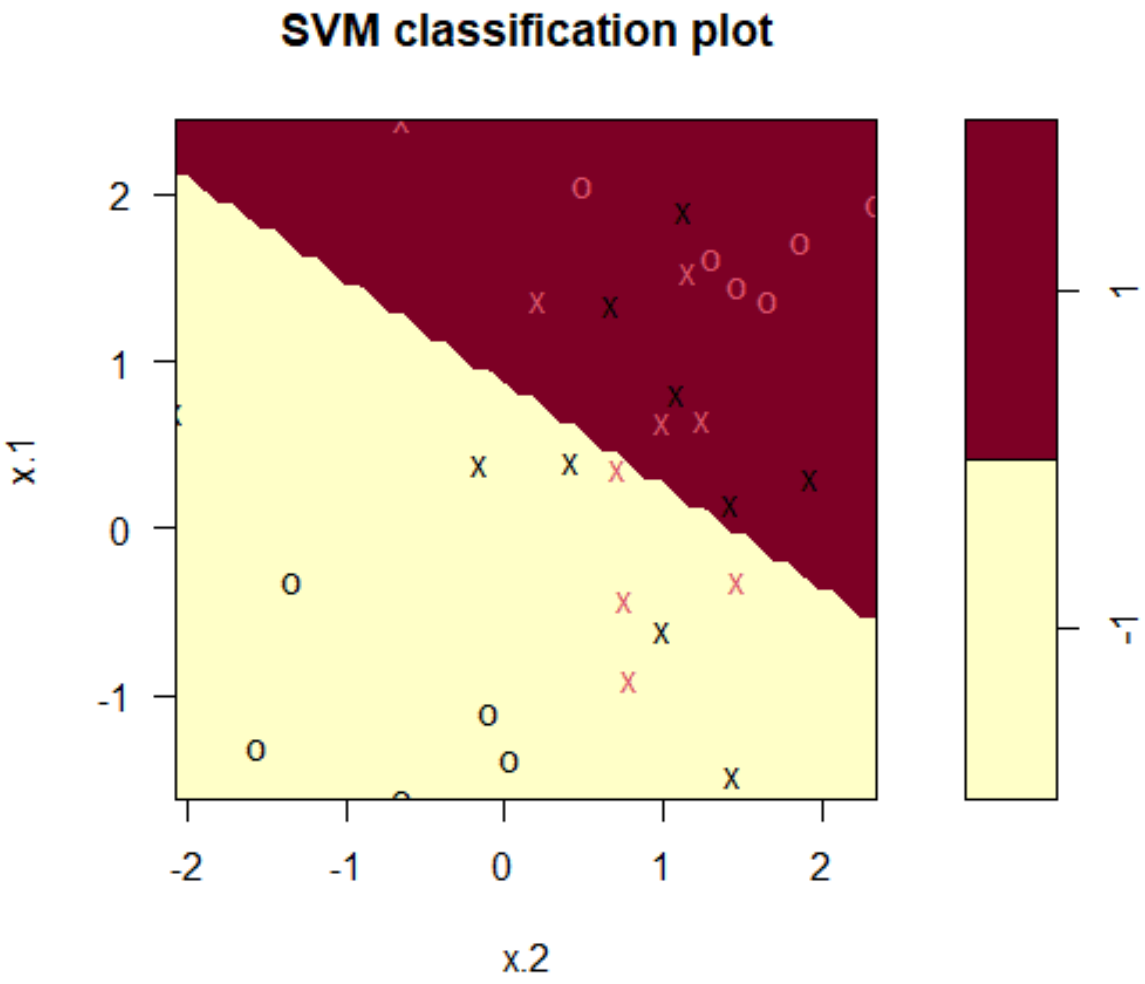


Figure 23: Imagen19

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

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

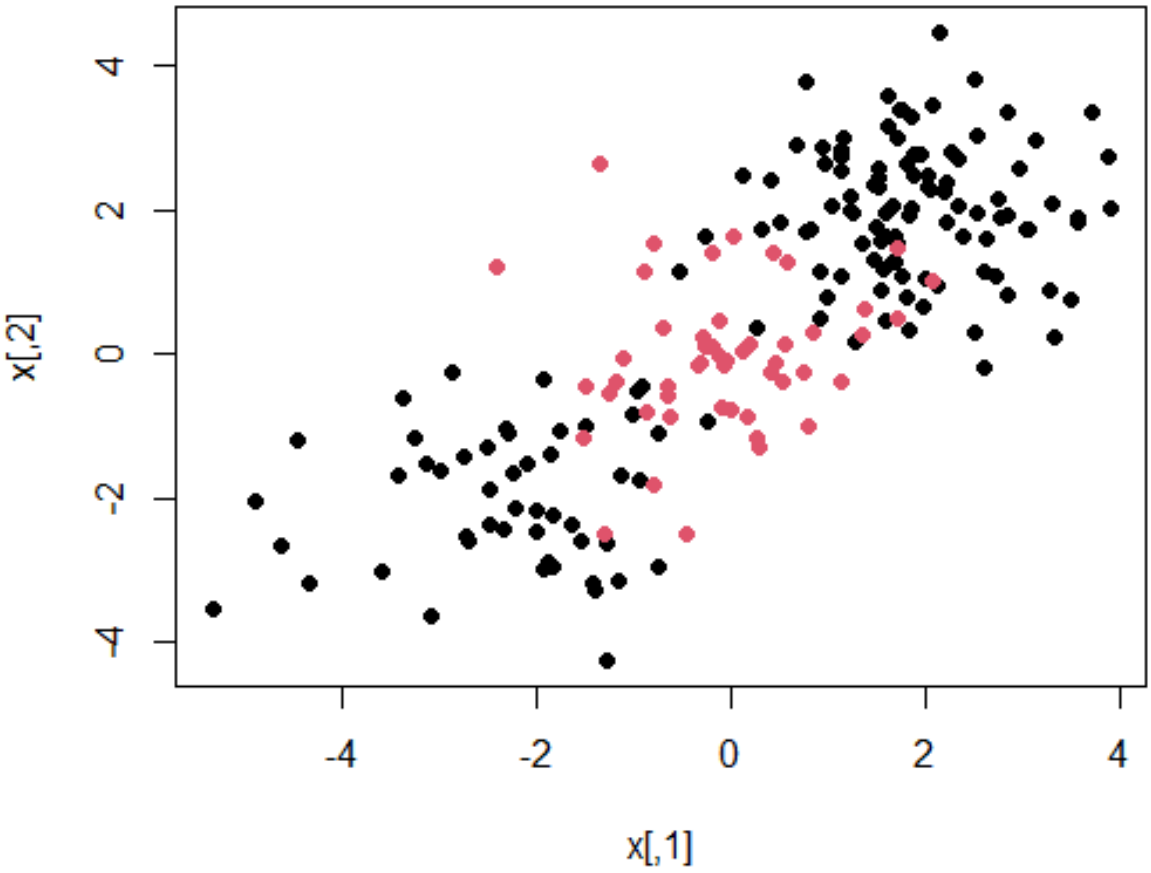


Figure 27: Imagen23

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

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 α_i , $i = 1, \cdots, n,$ uno por observación de entrenamiento.

2. Para estimar los parámetros $\alpha_1, \cdots, \alpha_n$ y β_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 α_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 α_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

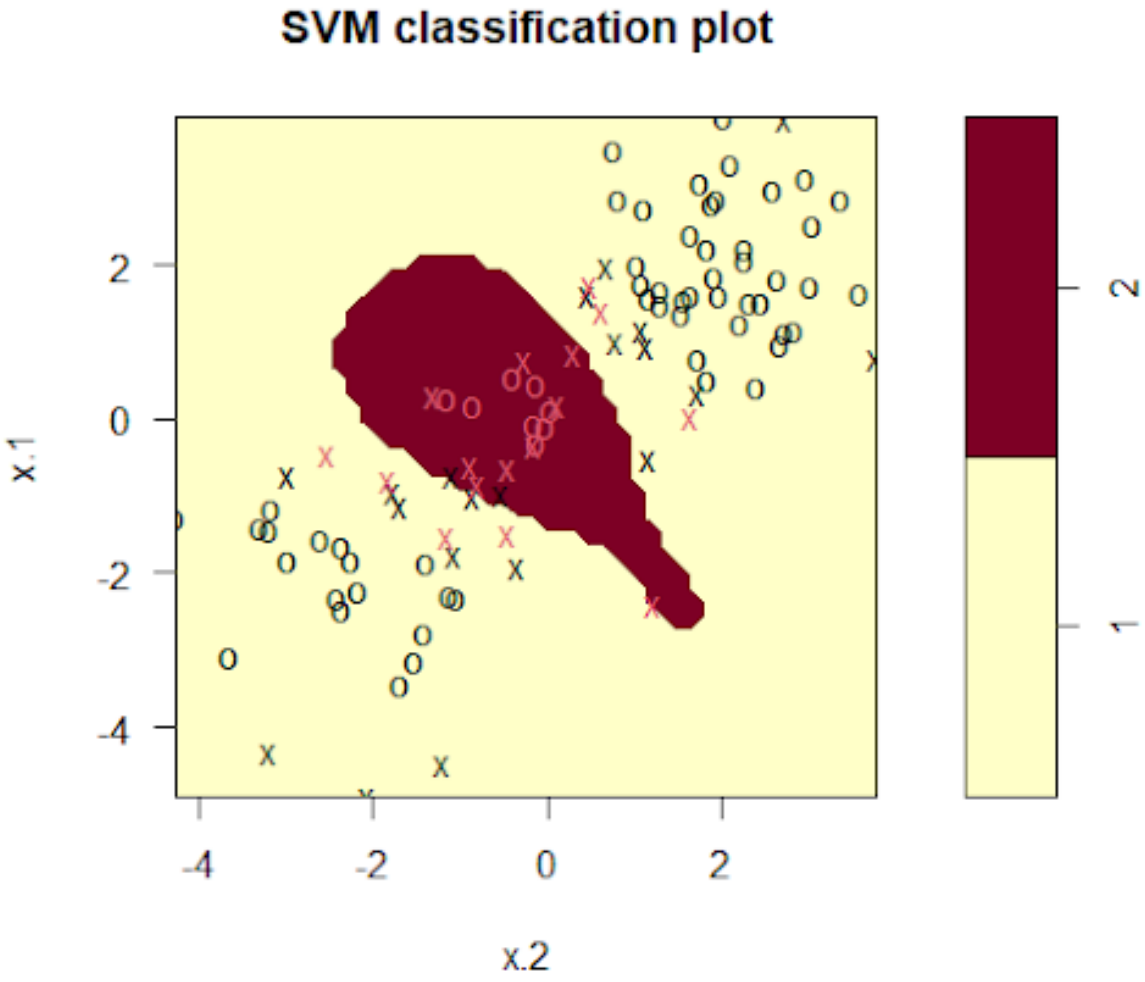


Figure 42: Imagen

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))
```

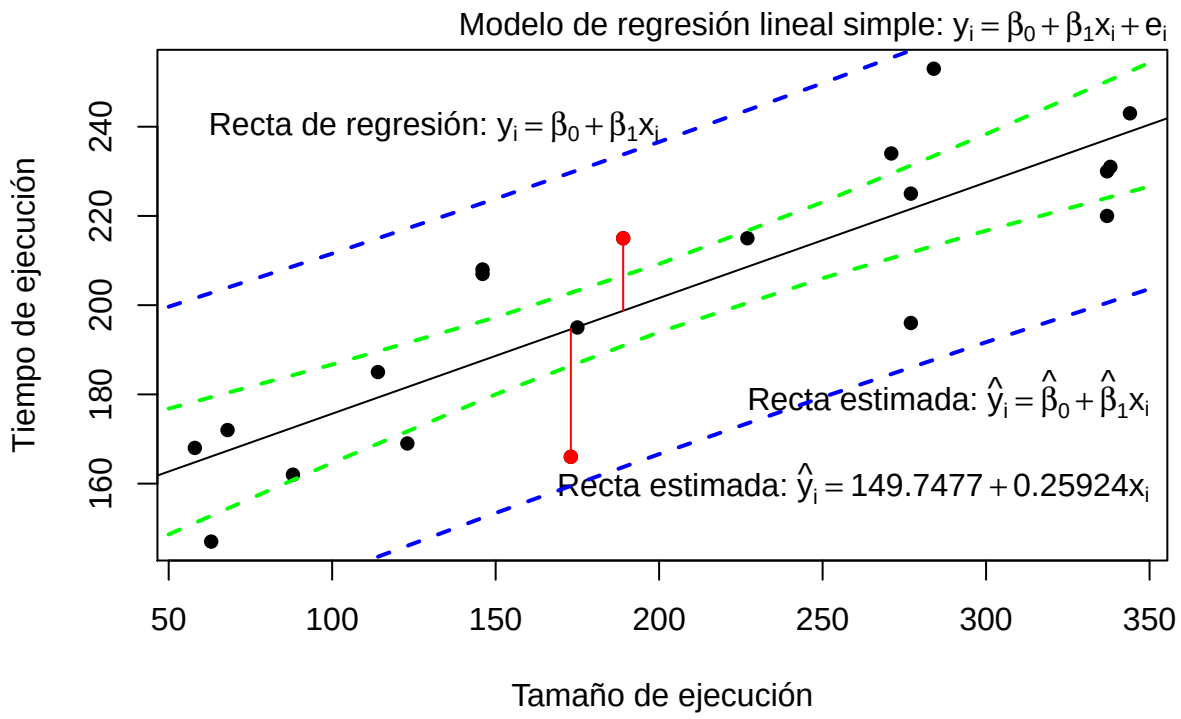
```
       fit      lwr      upr
1 162.7099 148.6204 176.7994
2 175.6720 164.6568 186.6872
3 188.6342 179.9969 197.2714
4 201.5963 193.9600 209.2326
5 214.5585 206.0455 223.0714
6 227.5206 216.7006 238.3407
7 240.4828 226.6220 254.3435
```

```
# 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
```

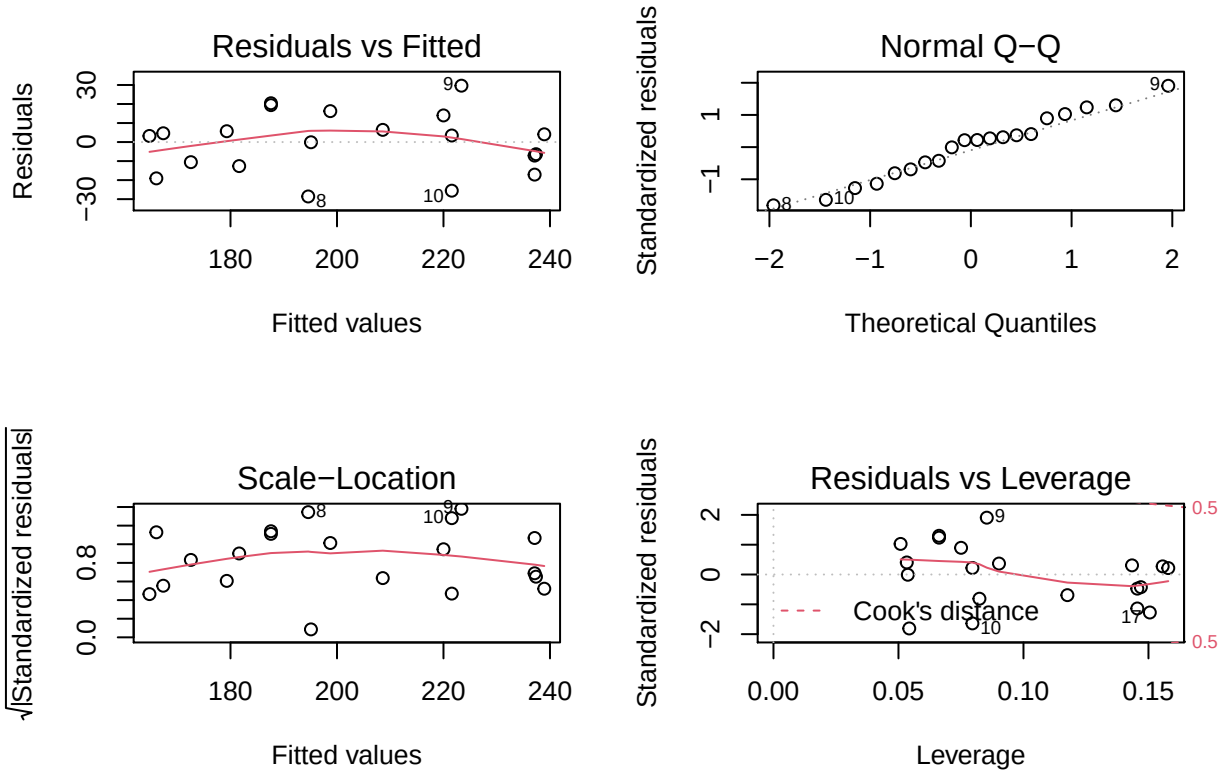



```r
# Note como los intervalos de confianza están contenidos dentro de los
# intervalos de predicción correspondientes
```

```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

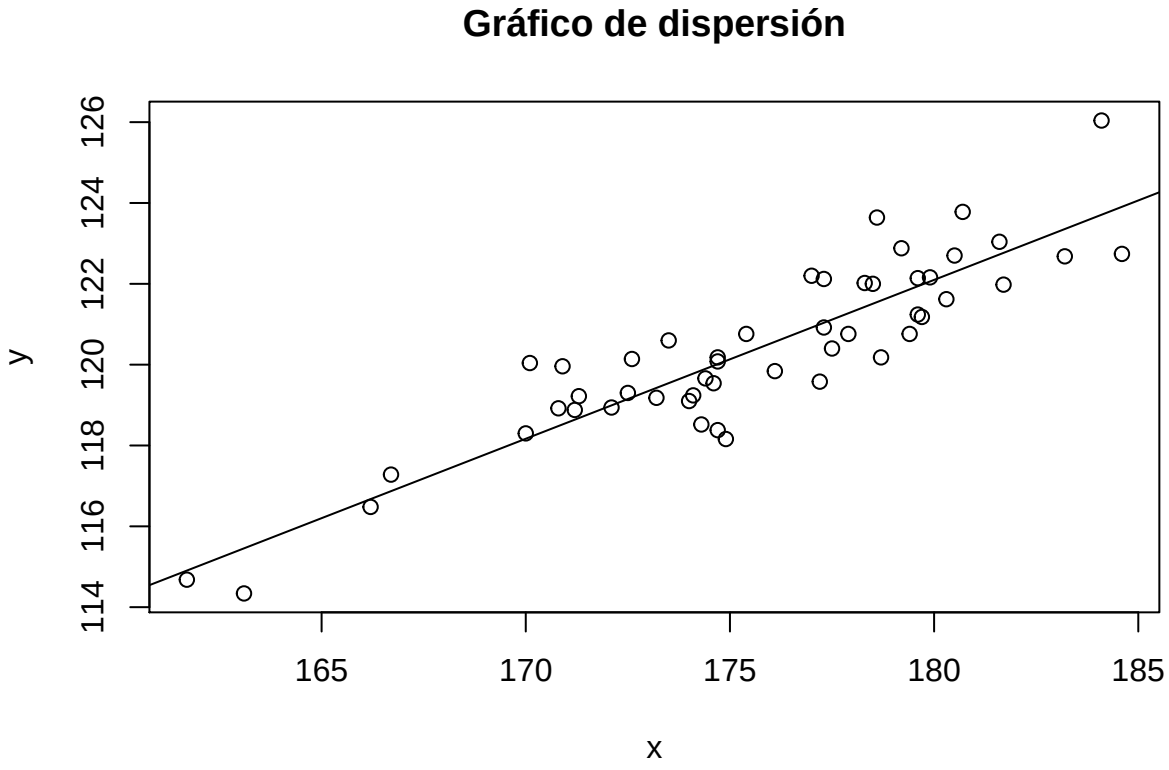
Se cree que entre las variables x y y del archivo csv adjunto, podría haber una relación más o menos lineal. Para tener más evidencia sobre esto lleve a cabo lo siguiente:

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.

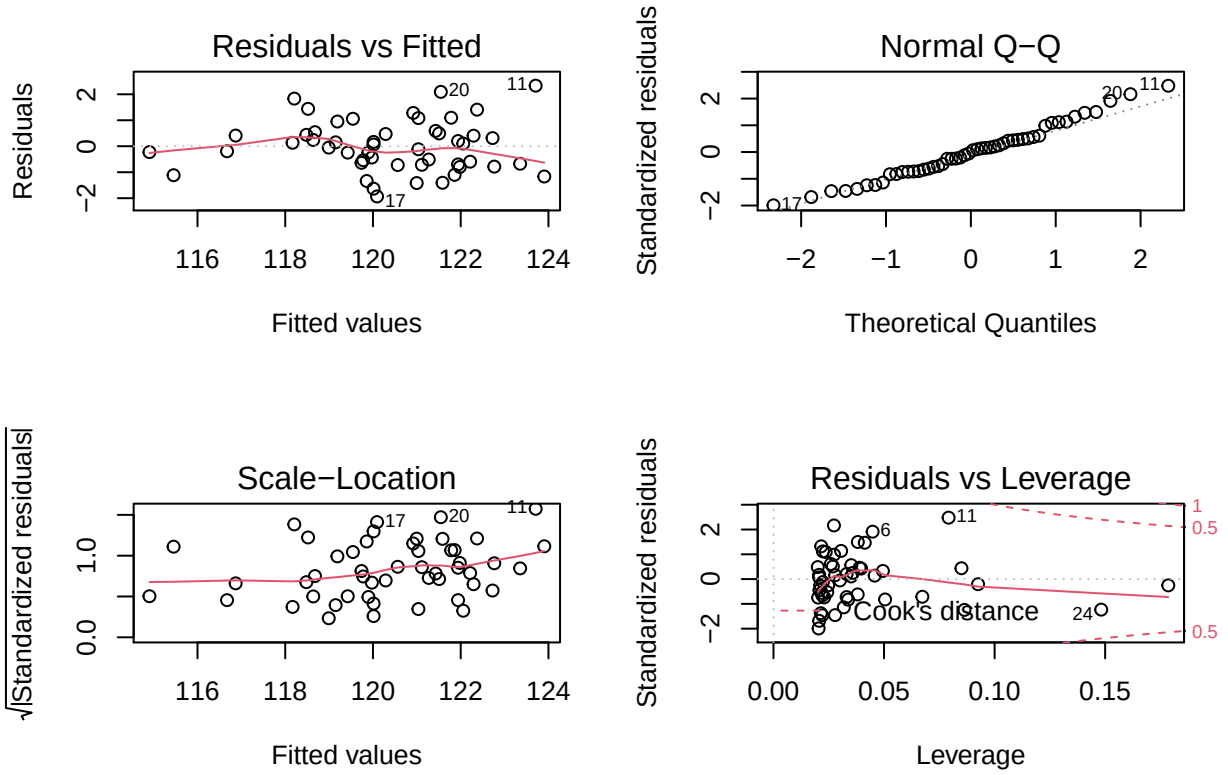
```r
# Reto 1. Regresión lineal simple

# 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)
```

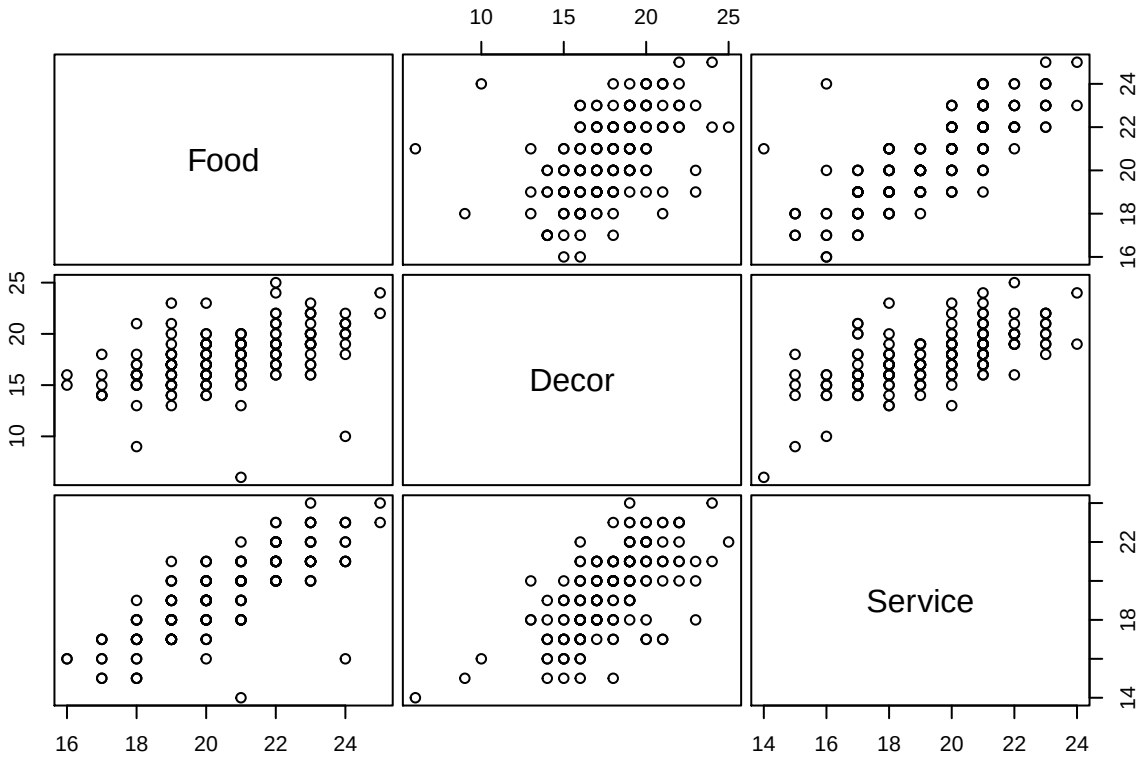
```
Call:
lm(formula = Price ~ Food + Decor + Service + East + Food:East +
    Decor:East + Service:East)

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)
```

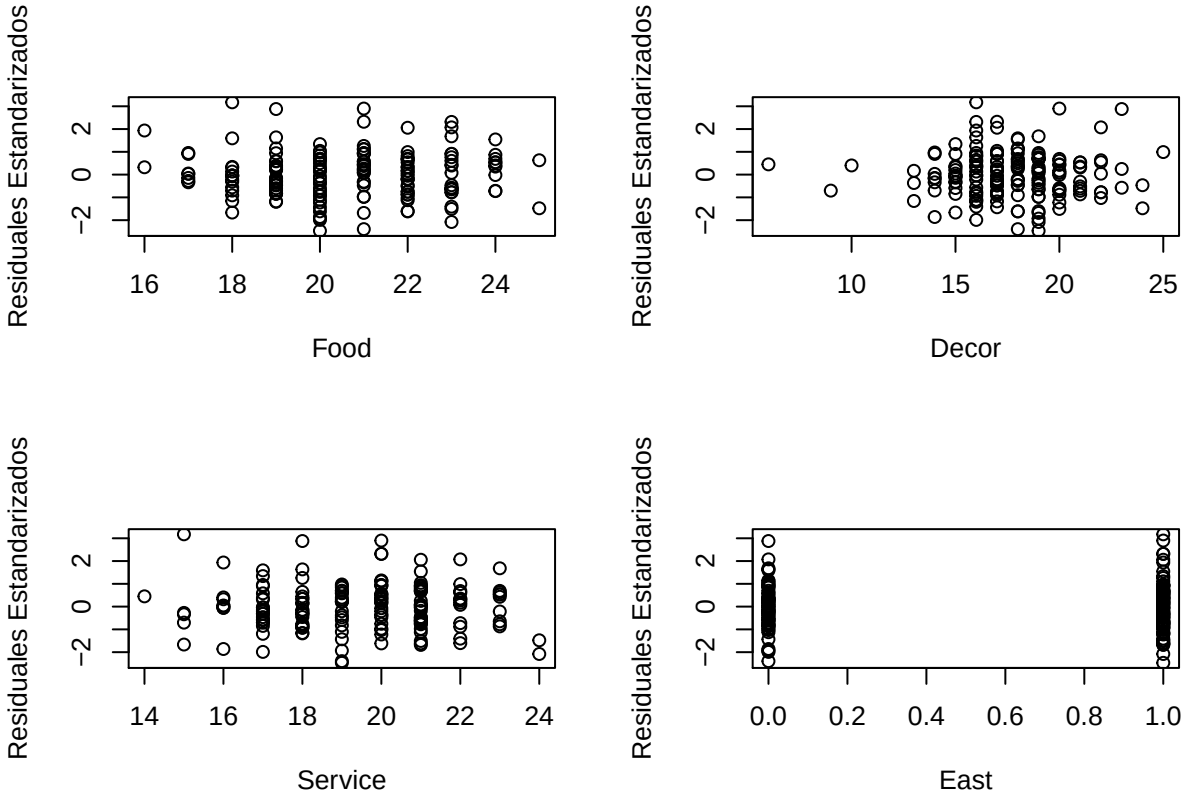



```r
# Acontinuación veremos gráficas de residuales estandarizados contra cada
# predictor. La naturaleza aleatoria de estas gráficas es un indicativo de
# que el modelo ajustado es un modelo válido para los datos.

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
```

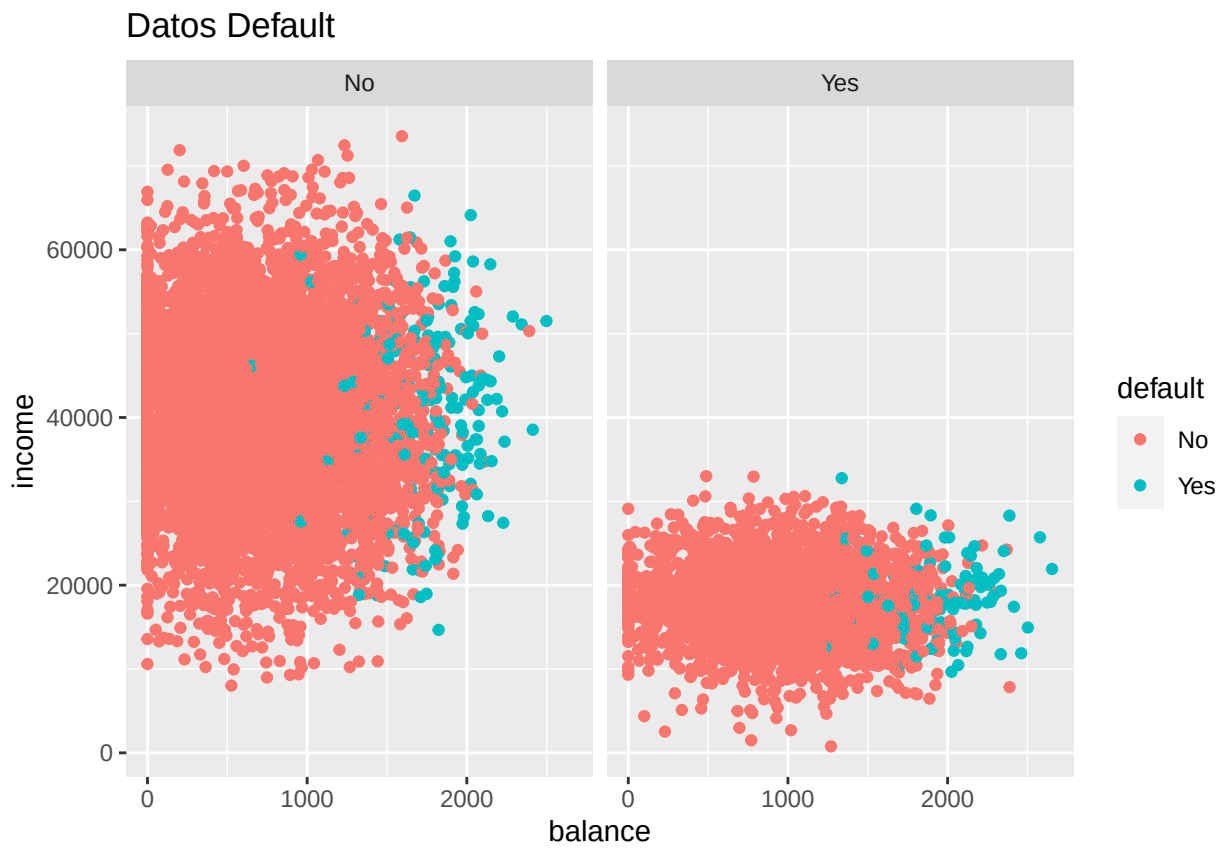
```r
dim(Default)
```

```
[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 ...
```

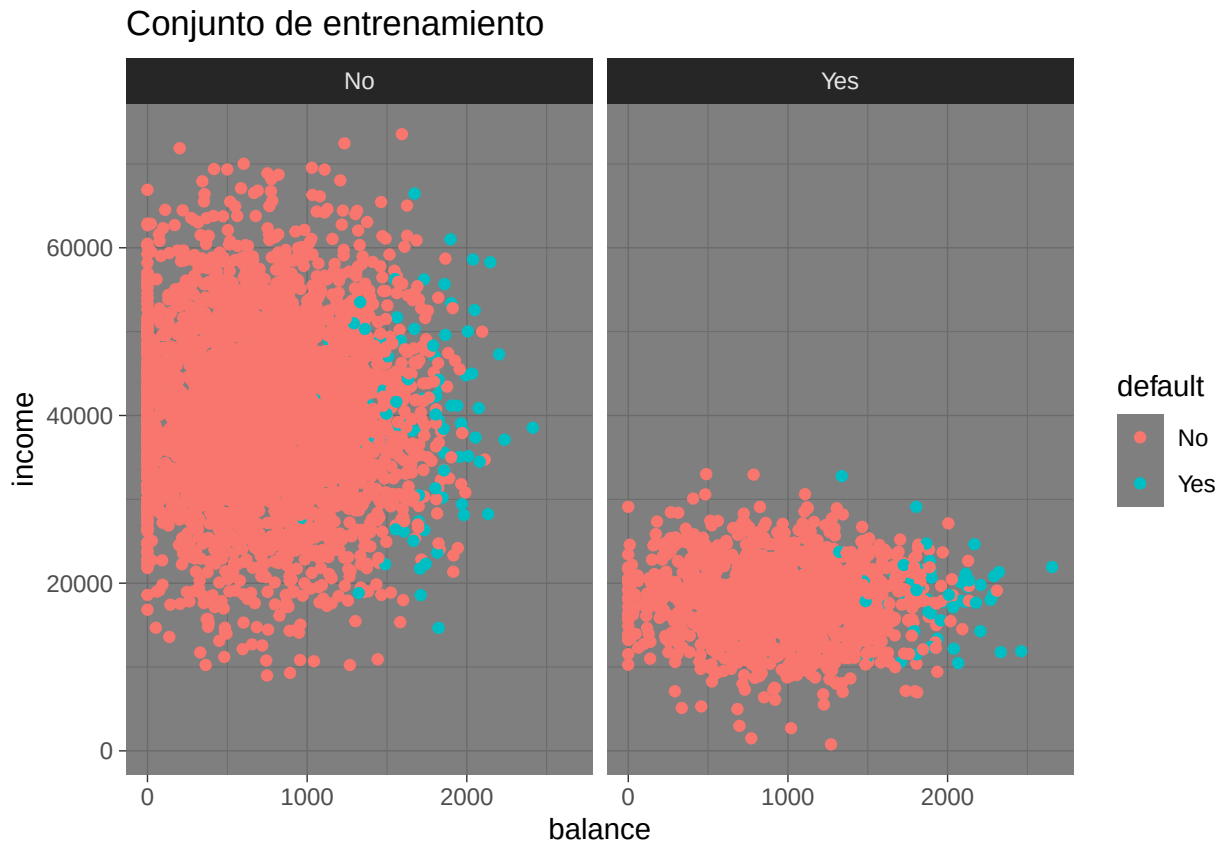
```r
# 2. Usando ggplot del paquete ggplot2, realicemos un gráfico de dispersión con la variable balance en

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



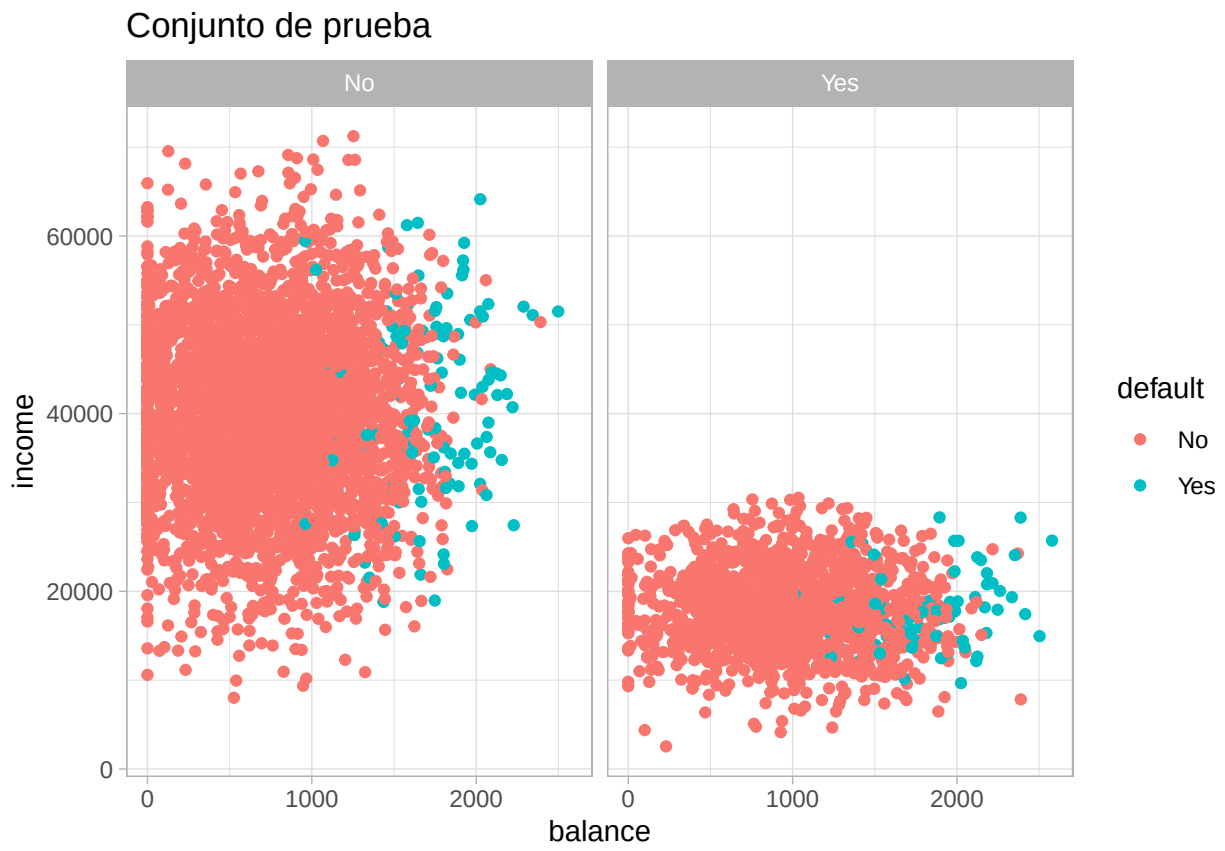

```
# 3. Generemos un vector de índices llamado train, tomando de manera aleatoria 5000 números de los prim

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):

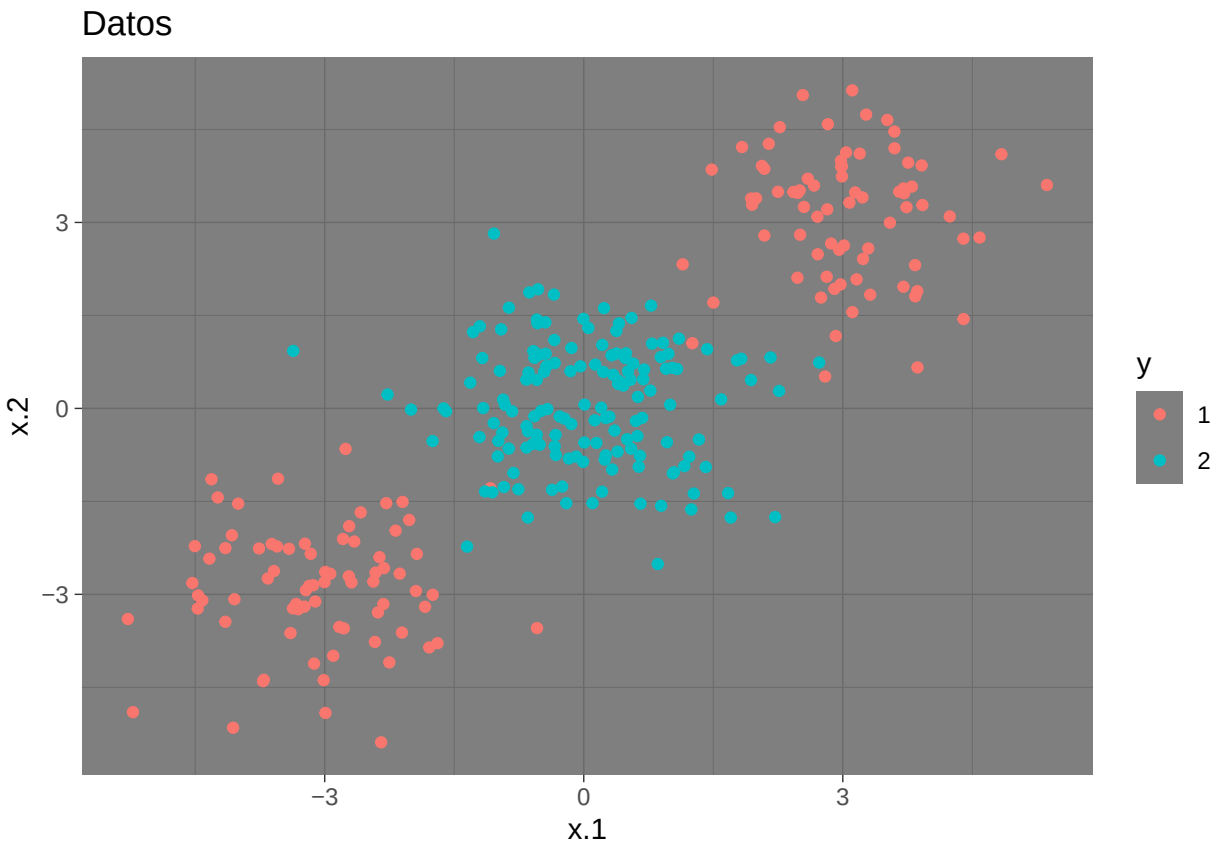
1. Carga los paquetes ggplot2 y e1071; observe algunas características del data frame con las funciones tail y dim. Obtenga el gráfico de dispersión de los datos diferenciando las dos clases.
2. Genera de manera aleatoria un vector de índices para filtrar un conjunto de entrenamiento a partir del conjunto 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)

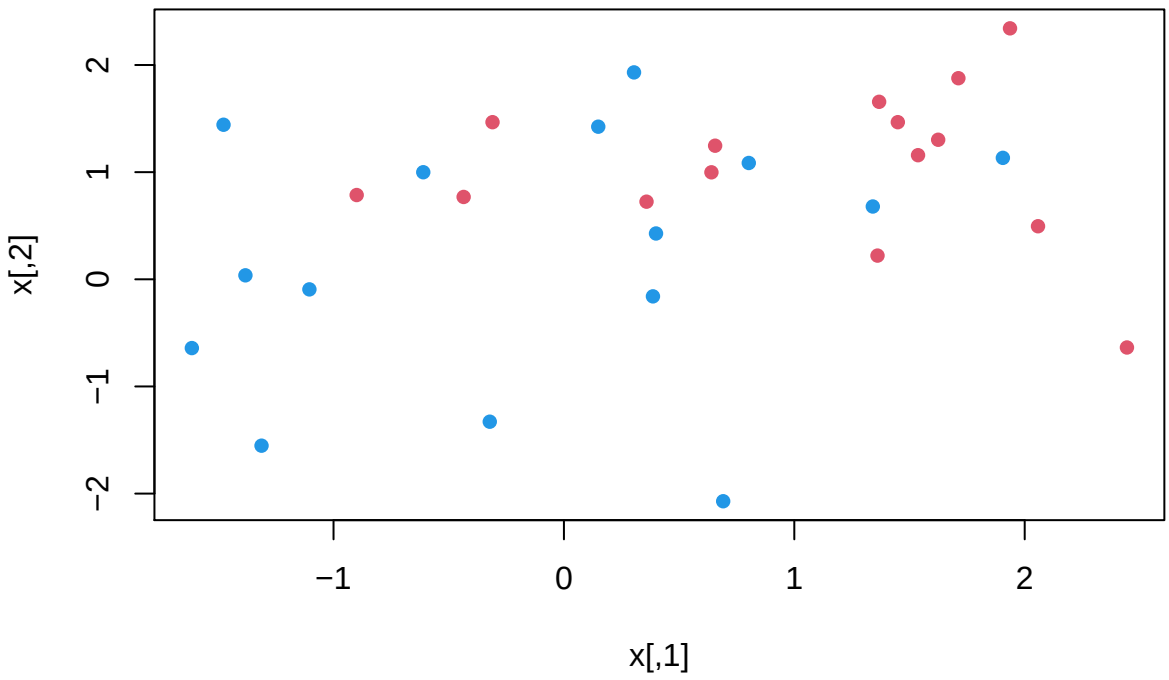
plot(svmfit, dat[train,]) summary(svmfit) Ajustamos una máquina de vectores de soporte con un kernel radial y valores de los parámetros gamma = 1 y cost = 1e5

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)
```

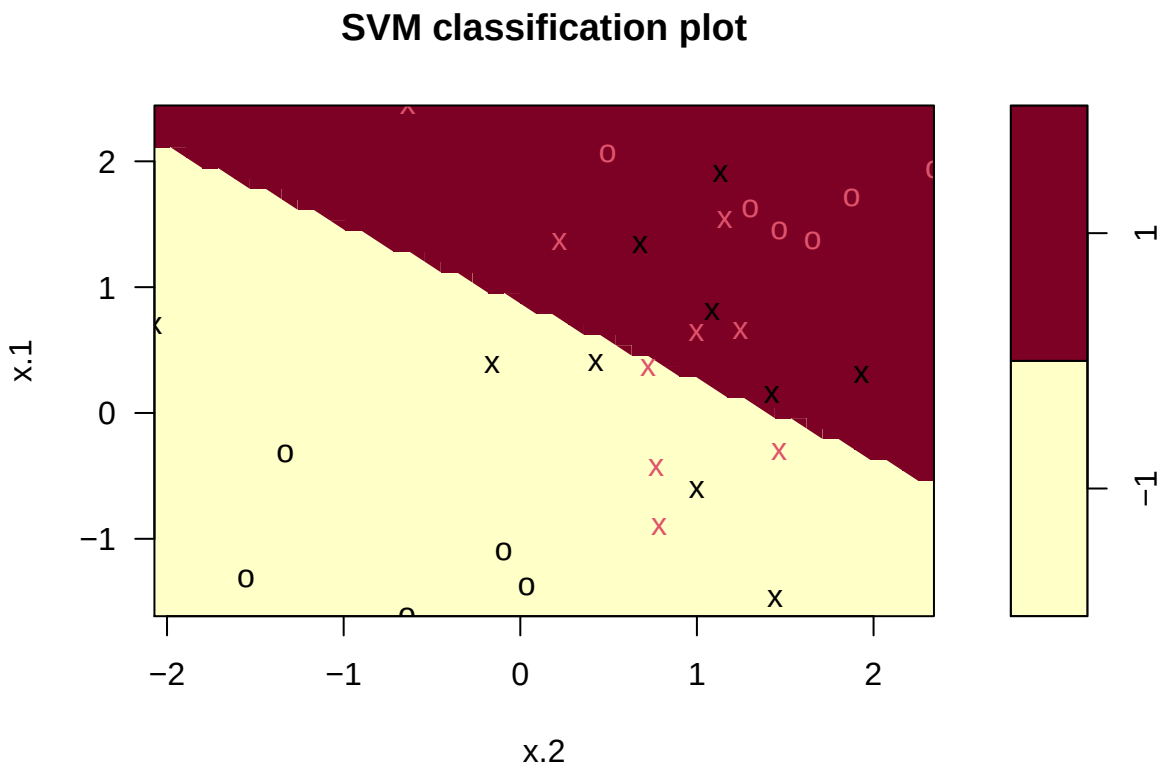



```
###

# 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
```

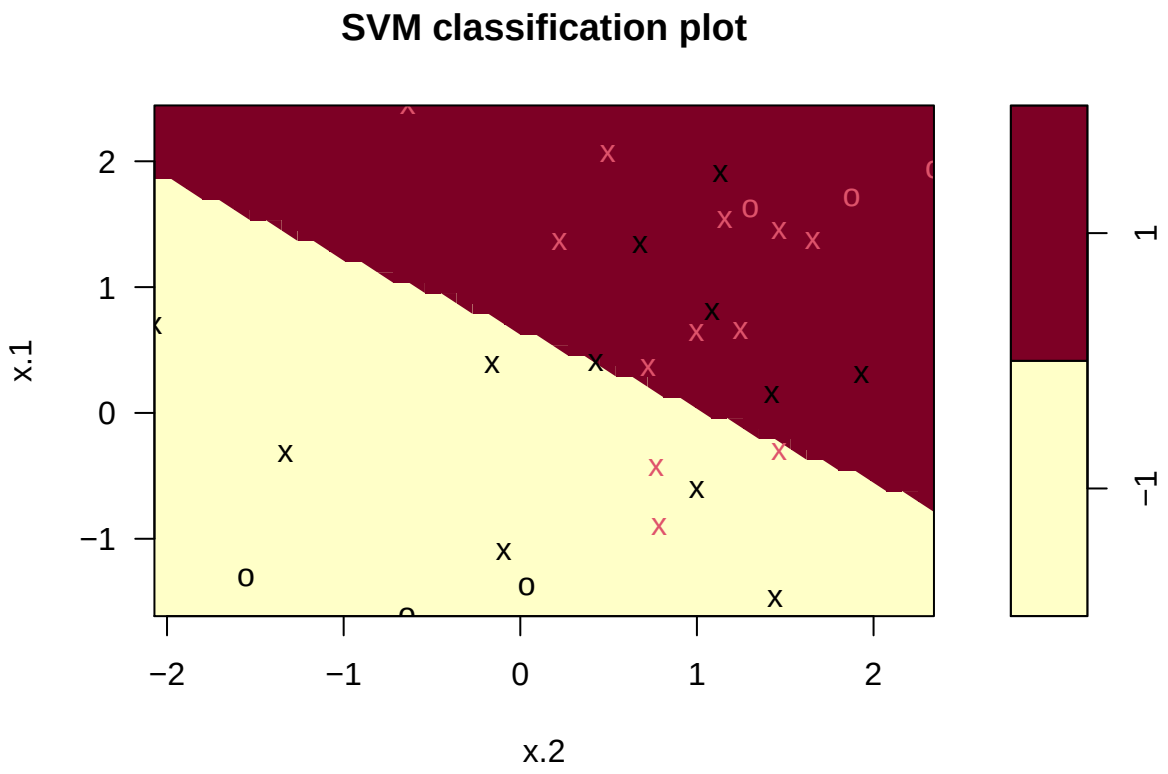
```
length(svmfit$index)
```

```
[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)
```

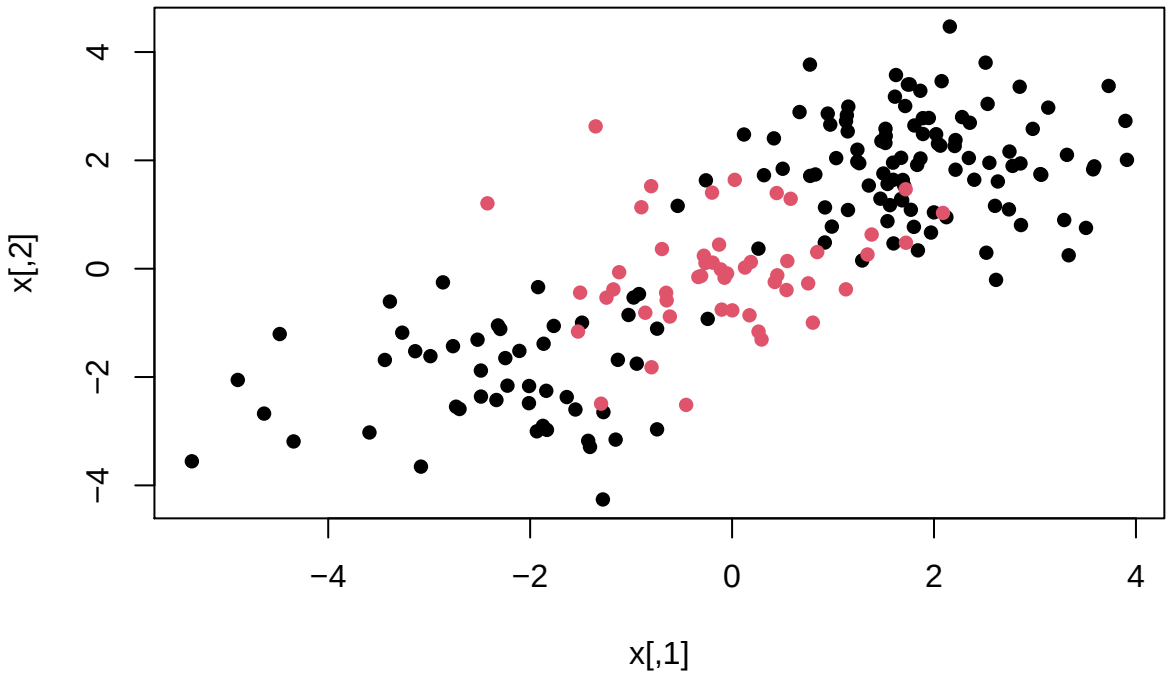
```
       truth
predict -1  1
     -1 11  5
     1   9 20
```

```
###

#### 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
```

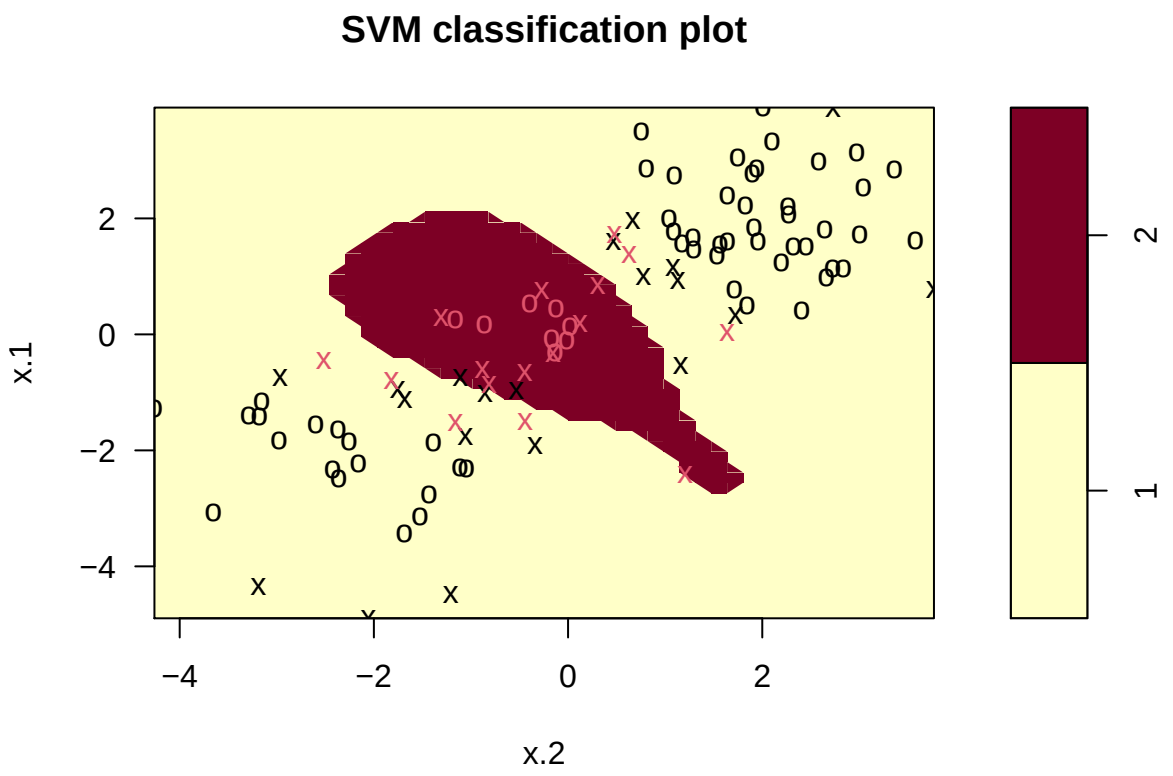
```
###


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, ])
```

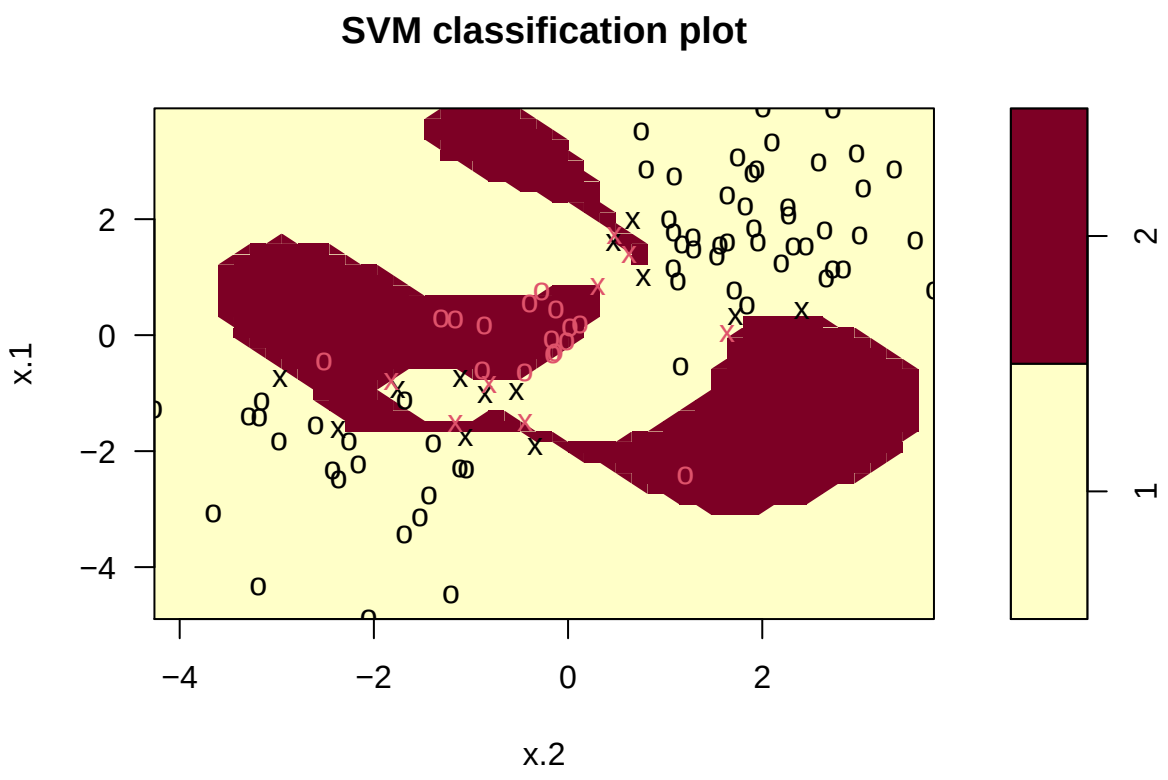



```
###

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
```

POSTWORK 5

```
# Postwork Sesión 5

# 1. A partir del conjunto de datos de soccer de la liga española
# de las temporadas 2017/2018, 2018/2019 y 2019/2020, creé el data
# frame `SmallData`, que contenga las columnas `date`, `home.team`,
# `home.score`, `away.team` y `away.score`; esto lo puede hacer con
# ayuda de la función `select` del paquete `dplyr`. Luego establezca
# un directorio de trabajo y con ayuda de la función `write.csv` guarde
# el data frame como un archivo csv con nombre *soccer.csv*.
# Puede colocar como argumento `row.names = FALSE` en `write.csv`.
# 2. Con la función `create.fbRanks.dataframes` del paquete
# `fbRanks` importe el archivo *soccer.csv* a `R` y al mismo
# tiempo asignelo a una variable llamada `listasoccer`. Se creará
# una lista con los elementos `scores` y `teams` que son data frames
# listos para la función `rank.teams`. Asigne estos data frames a
# variables llamadas `anotaciones` y `equipos`.
# 3. Con ayuda de la función `unique` creé un vector de
# fechas (`fecha`) que no se repitan y que correspondan
# a las fechas en las que se jugaron partidos. Creé una
# variable llamada `n` que contenga el número de fechas diferentes.
# Posteriormente, con la función `rank.teams` y usando como argumentos
# los data frames `anotaciones` y `equipos`, creé un ranking de equipos
# usando unicamente datos desde la fecha inicial y hasta la penúltima
# fecha en la que se jugaron partidos, estas fechas las deberá
# especificar en `max.date` y `min.date`. Guarde los resultados
# con el nombre `ranking`.
# 4. Finalmente estime las probabilidades de los eventos,
# el equipo de casa gana, el equipo visitante gana o
# el resultado es un empate para los partidos que se
# jugaron en la última fecha del vector de fechas `fecha`.
# Esto lo puede hacer con ayuda de la función `predict` y
# usando como argumentos `ranking` y `fecha[n]` que deberá
# especificar en `date`.

####

# **Solución**
```

```r
# Lo primero que haremos es cargar los paquetes que
# usaremos más adelante. Usamos las funciones 'suppressWarnings'
# y 'supperssMessages' para que no se impriman mensajes ni
# advertencias al cargar el paquete.

suppressWarnings(suppressMessages(library(dplyr)))
suppressWarnings(suppressMessages(library(fbRanks)))

# Comenzamos importando los datos que se encuentran en archivos csv a 'R'

url1718 <- "https://www.football-data.co.uk/mmz4281/1718/SP1.csv"
url1819 <- "https://www.football-data.co.uk/mmz4281/1819/SP1.csv"
url1920 <- "https://www.football-data.co.uk/mmz4281/1920/SP1.csv"
d1718 <- read.csv(file = url1718) # Importación de los datos a R
d1819 <- read.csv(file = url1819)
d1920 <- read.csv(file = url1920)

# Obtenemos una mejor idea de los datos que se
# encuentran en los data frames con las funciones 'str',
# 'head', 'View' y 'summary'

str(d1718); str(d1819); str(d1920)
```

```
'data.frame':   380 obs. of  64 variables:
 $ Div       : chr  "SP1" "SP1" "SP1" "SP1" ...
 $ Date      : chr  "18/08/17" "18/08/17" "19/08/17" "19/08/17" ...
 $ HomeTeam  : chr  "Leganes" "Valencia" "Celta" "Girona" ...
 $ AwayTeam  : chr  "Alaves" "Las Palmas" "Sociedad" "Ath Madrid" ...
 $ FTHG      : int  1 1 2 2 1 0 2 0 1 0 ...
 $ FTAG      : int  0 0 3 2 1 0 0 3 0 1 ...
 $ FTR       : chr  "H" "H" "A" "D" ...
 $ HTHG      : int  1 1 1 2 1 0 2 0 0 0 ...
 $ HTAG      : int  0 0 1 0 1 0 0 2 0 0 ...
 $ HTR       : chr  "H" "H" "D" "H" ...
 $ HS        : int  16 22 16 13 9 12 15 12 14 10 ...
 $ AS        : int  6 5 13 9 9 8 3 16 9 13 ...
 $ HST       : int  9 6 5 6 4 2 2 6 3 4 ...
 $ AST       : int  3 4 6 3 6 2 0 8 1 6 ...
 $ HF        : int  14 25 12 15 14 16 16 16 18 16 ...
 $ AF        : int  18 13 11 15 12 15 15 12 14 15 ...
 $ HC        : int  4 5 5 6 7 7 8 4 11 3 ...
 $ AC        : int  2 2 4 0 3 6 0 4 6 7 ...
 $ HY        : int  0 3 3 2 2 1 2 5 1 2 ...
 $ AY        : int  1 3 1 4 4 3 1 1 3 3 ...
 $ HR        : int  0 0 0 0 1 0 0 0 0 0 ...
 $ AR        : int  0 1 0 1 0 1 0 1 0 0 ...
 $ B365H     : num  2.05 1.75 2.38 8 1.62 1.5 1.17 9.5 3.25 2.1 ...
 $ B365D     : num  3.2 3.8 3.25 4.33 4 4 8 5.75 3.25 3.3 ...
 $ B365A     : num  4.1 4.5 3.2 1.45 5.5 7.5 15 1.3 2.3 3.7 ...
 $ BWH       : num  2.05 1.75 2.4 7.5 1.62 1.48 1.18 9.25 3.25 2.15 ...
 $ BWD       : num  3.1 3.9 3.3 4.33 3.9 4.25 7.5 5.75 3.2 3.3 ...
 $ BWA       : num  4.1 4.6 3 1.45 5.75 7 14.5 1.3 2.3 3.5 ...
 $ IWH       : num  2.1 1.75 2.5 7.2 1.55 1.5 1.17 7.5 3.3 2.1 ...
```

```
 $ IWD       : num  3.4 3.6 3.3 4.4 4 4.2 7.5 5.5 3.35 3.4 ...
 $ IWA       : num  3.5 4.8 2.85 1.45 6.2 6.5 15 1.35 2.2 3.5 ...
 $ LBH       : num  2.05 1.75 2.35 7.5 1.6 1.5 1.2 9.5 3.25 2.1 ...
 $ LBD       : num  3 3.8 3.25 4 3.9 4 6.5 5.25 3.1 3.1 ...
 $ LBA       : num  4.2 4.33 3 1.5 5.5 7 15 1.3 2.3 3.4 ...
 $ PSH       : num  2.03 1.78 2.44 8.36 1.62 ...
 $ PSD       : num  3.25 4.01 3.4 4.38 4.17 4.37 7.35 5.79 3.24 3.36 ...
 $ PSA       : num  4.52 4.83 3.16 1.49 6.18 7.31 15.5 1.33 2.36 3.49 ...
 $ WHH       : num  2.05 1.8 2.4 8 1.67 1.5 1.22 11 3.1 2.2 ...
 $ WHD       : num  3.1 3.75 3.4 4.2 3.6 4 6 4.5 3.1 3.3 ...
 $ WHA       : num  4 4.2 2.9 1.44 5.5 7 13 1.33 2.4 3.3 ...
 $ VCH       : num  2.05 1.8 2.4 7.5 1.65 1.5 1.2 9.5 3.25 2.15 ...
 $ VCD       : num  3.2 4 3.4 4.3 4 4.2 7 5.75 3.25 3.3 ...
 $ VCA       : num  4.4 4.6 3.13 1.5 5.75 7 13 1.3 2.3 3.5 ...
 $ Bb1X2     : int  35 35 35 35 35 34 35 35 34 34 ...
 $ BbMxH     : num  2.12 1.83 2.5 8.36 1.69 ...
 $ BbAvH     : num  2.03 1.77 2.39 7.53 1.63 1.5 1.19 9.68 3.26 2.18 ...
 $ BbMxD     : num  3.4 4.04 3.5 4.4 4.17 4.4 8 5.86 3.35 3.4 ...
 $ BbAvD     : num  3.15 3.86 3.32 4.17 3.93 4.17 7.11 5.44 3.17 3.26 ...
 $ BbMxA     : num  4.52 4.83 3.2 1.51 6.2 7.5 17 1.35 2.4 3.7 ...
 $ BbAvA     : num  4.17 4.46 3.01 1.48 5.58 ...
 $ BbOU      : int  31 33 34 34 33 32 27 27 32 32 ...
 $ BbMx.2.5  : num  2.84 1.69 2.03 2.2 1.81 2.01 1.44 1.5 2.42 2.25 ...
 $ BbAv.2.5  : num  2.68 1.64 1.98 2.11 1.75 1.94 1.4 1.46 2.36 2.14 ...
 $ BbMx.2.5.1: num  1.53 2.4 1.9 1.8 2.14 1.96 3.1 2.95 1.63 1.76 ...
 $ BbAv.2.5.1: num  1.46 2.27 1.84 1.74 2.09 1.87 2.88 2.64 1.58 1.7 ...
 $ BbAH      : int  18 16 18 16 16 17 17 16 15 17 ...
 $ BbAHh     : num  -0.5 -0.75 -0.25 1.25 -1 -1 -2 1.5 0.25 -0.25 ...
 $ BbMxAHH   : num  2.07 2.05 2.08 1.77 2.12 1.9 2.05 2.03 1.93 1.92 ...
 $ BbAvAHH   : num  2.03 1.97 2.05 1.75 2.06 1.86 2 1.98 1.89 1.88 ...
 $ BbMxAHA   : num  1.9 1.96 1.87 2.25 1.86 2.05 1.91 1.95 2.03 2.04 ...
 $ BbAvAHA   : num  1.86 1.91 1.83 2.16 1.82 2.01 1.86 1.89 1.98 1.99 ...
 $ PSCH      : num  1.98 1.78 2.12 6.93 1.64 1.53 1.2 12.4 3.31 2.2 ...
 $ PSCD      : num  3.35 4.24 3.53 3.83 4.18 4.48 8.25 7 3.32 3.27 ...
 $ PSCA      : num  4.63 4.43 3.74 1.63 5.82 6.91 15.2 1.26 2.4 3.85 ...

'data.frame':   380 obs. of  61 variables:
 $ Div       : chr  "SP1" "SP1" "SP1" "SP1" ...
 $ Date      : chr  "17/08/2018" "17/08/2018" "18/08/2018" "18/08/2018" ...
 $ HomeTeam  : chr  "Betis" "Girona" "Barcelona" "Celta" ...
 $ AwayTeam  : chr  "Levante" "Valladolid" "Alaves" "Espanol" ...
 $ FTHG      : int  0 0 3 1 1 1 2 1 2 1 ...
 $ FTAG      : int  3 0 0 1 2 2 0 4 1 1 ...
 $ FTR       : chr  "A" "D" "H" "D" ...
 $ HTHG      : int  0 0 0 0 1 0 1 0 1 0 ...
 $ HTAG      : int  1 0 0 1 1 2 0 3 1 1 ...
 $ HTR       : chr  "A" "D" "D" "A" ...
 $ HS        : int  22 13 25 12 16 18 10 13 17 13 ...
 $ AS        : int  6 2 3 14 8 8 4 17 12 9 ...
 $ HST       : int  8 1 9 2 7 6 3 2 5 4 ...
 $ AST       : int  4 1 0 5 4 6 1 8 2 3 ...
 $ HF        : int  10 21 6 13 16 12 11 6 12 10 ...
 $ AF        : int  10 20 13 14 10 13 27 15 13 15 ...
 $ HC        : int  5 3 7 8 4 7 3 2 6 4 ...
```

```
 $ AC        : int  3 2 1 7 6 0 0 6 2 10 ...
 $ HY        : int  0 1 0 3 2 1 1 1 4 2 ...
 $ AY        : int  2 1 2 2 3 1 7 0 5 3 ...
 $ HR        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ AR        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ B365H     : num  1.66 1.75 1.11 1.85 2.04 1.66 1.2 3.25 1.75 3 ...
 $ B365D     : num  4 3.6 10 3.5 3.4 3.75 7 3.6 3.3 3.2 ...
 $ B365A     : num  5 5 21 4.5 3.8 5.5 13 2.14 5.5 2.5 ...
 $ BWH       : num  1.7 1.75 1.11 1.91 2.05 1.7 1.18 3.5 1.78 2.85 ...
 $ BWD       : num  3.7 3.5 10 3.4 3.3 3.7 7.25 3.5 3.5 3.25 ...
 $ BWA       : num  5.25 5.25 20 4.25 3.9 5.25 16 2.1 5 2.55 ...
 $ IWH       : num  1.75 1.8 1.12 1.9 2 1.7 1.2 3.5 1.85 2.85 ...
 $ IWD       : num  3.6 3.6 9 3.5 3.4 3.75 6.5 3.4 3.5 3.2 ...
 $ IWA       : num  4.9 4.5 20 4.1 3.8 5 15 2.1 4.4 2.55 ...
 $ PSH       : num  1.69 1.8 1.11 1.93 2.06 1.72 1.2 3.46 1.79 3.12 ...
 $ PSD       : num  4.19 3.7 11.27 3.64 3.51 ...
 $ PSA       : num  5.11 4.99 25.4 4.27 3.91 ...
 $ WHH       : num  1.67 1.75 1.08 1.91 2.05 1.73 1.22 3.3 1.8 3 ...
 $ WHD       : num  3.9 3.6 9 3.5 3.3 3.6 6 3.7 3.4 3.2 ...
 $ WHA       : num  4.75 4.6 29 4 3.6 4.75 13 2.05 4.75 2.4 ...
 $ VCH       : num  1.67 1.8 1.1 1.93 2.05 1.7 1.2 3.4 1.8 3 ...
 $ VCD       : num  4.2 3.7 10.5 3.5 3.5 3.8 7 3.6 3.4 3.2 ...
 $ VCA       : num  5.2 4.8 34 4.4 3.9 5 13 2.1 5 2.45 ...
 $ Bb1X2     : int  40 40 40 38 40 40 39 40 40 39 ...
 $ BbMxH     : num  1.75 1.85 1.13 1.97 2.11 1.76 1.24 3.53 1.85 3.12 ...
 $ BbAvH     : num  1.68 1.78 1.1 1.9 2.03 1.7 1.21 3.38 1.78 2.99 ...
 $ BbMxD     : num  4.25 3.83 11.5 3.73 3.62 3.93 7.36 3.75 3.64 3.29 ...
 $ BbAvD     : num  4 3.6 9.82 3.53 3.43 3.77 6.66 3.56 3.43 3.14 ...
 $ BbMxA     : num  5.25 5.27 41 4.5 3.93 ...
 $ BbAvA     : num  4.95 4.79 25.67 4.2 3.76 ...
 $ BbOU      : int  38 38 32 36 37 37 33 37 36 36 ...
 $ BbMx.2.5  : num  1.82 2.21 1.39 2.13 2.05 1.95 1.5 1.83 2.49 2.45 ...
 $ BbAv.2.5  : num  1.76 2.13 1.34 2.06 1.99 1.88 1.45 1.76 2.35 2.33 ...
 $ BbMx.2.5.1: num  2.15 1.78 3.4 1.84 1.88 1.98 2.75 2.13 1.64 1.65 ...
 $ BbAv.2.5.1: num  2.06 1.71 3.18 1.76 1.81 1.91 2.66 2.04 1.58 1.59 ...
 $ BbAH      : int  20 20 19 18 18 19 19 19 18 17 ...
 $ BbAHh     : num  -0.75 -0.75 -2.5 -0.75 -0.25 -0.75 -1.75 0.25 -0.75 0.25 ...
 $ BbMxAHH   : num  1.89 2.06 1.95 2.26 1.76 1.96 1.85 2.08 2.11 1.82 ...
 $ BbAvAHH   : num  1.85 2.01 1.91 2.18 1.74 1.91 1.8 2.03 2.04 1.75 ...
 $ BbMxAHA   : num  2.07 1.9 2 1.74 2.23 2.01 2.15 1.86 1.86 2.23 ...
 $ BbAvAHA   : num  2 1.85 1.95 1.71 2.14 1.94 2.07 1.83 1.82 2.12 ...
 $ PSCH      : num  1.59 1.76 1.1 2.18 2.32 1.77 1.19 4.57 1.69 3.55 ...
 $ PSCD      : num  4.42 3.57 11.85 3.26 3.21 ...
 $ PSCA      : num  5.89 5.62 32.17 3.85 3.53 ...

'data.frame':   380 obs. of  105 variables:
 $ Div        : chr  "SP1" "SP1" "SP1" "SP1" ...
 $ Date       : chr  "16/08/2019" "17/08/2019" "17/08/2019" "17/08/2019" ...
 $ Time       : chr  "20:00" "16:00" "18:00" "19:00" ...
 $ HomeTeam   : chr  "Ath Bilbao" "Celta" "Valencia" "Mallorca" ...
 $ AwayTeam   : chr  "Barcelona" "Real Madrid" "Sociedad" "Eibar" ...
 $ FTHG       : int  1 1 1 2 0 4 1 0 1 1 ...
 $ FTAG       : int  0 3 1 1 1 4 0 2 2 0 ...
 $ FTR        : chr  "H" "A" "D" "H" ...
```

```
$ HTHG       : int  0 0 0 1 0 1 0 0 0 1 ...
$ HTAG       : int  0 1 0 0 0 1 0 1 0 0 ...
$ HTR        : chr  "D" "A" "D" "H" ...
$ HS         : int  11 7 14 16 13 12 9 7 13 5 ...
$ AS         : int  11 17 12 11 4 14 16 12 14 6 ...
$ HST        : int  5 4 6 4 2 7 2 2 4 5 ...
$ AST        : int  2 11 3 5 2 7 4 4 3 0 ...
$ HF         : int  14 17 13 13 17 10 18 11 11 19 ...
$ AF         : int  9 12 14 14 11 16 15 17 19 22 ...
$ HC         : int  3 6 3 9 8 2 2 8 6 3 ...
$ AC         : int  8 4 3 3 0 7 9 4 1 4 ...
$ HY         : int  1 5 4 2 1 3 2 2 2 3 ...
$ AY         : int  1 2 4 3 4 1 1 2 6 4 ...
$ HR         : int  0 0 1 0 1 0 0 0 1 1 ...
$ AR         : int  0 1 0 0 0 0 0 0 0 1 ...
$ B365H      : num  5.25 4.75 1.66 2.8 2 1.6 2.15 3.2 1.66 1.44 ...
$ B365D      : num  3.8 4.2 3.75 3.2 3.2 3.8 3.2 3.3 3.75 4.33 ...
$ B365A      : num  1.65 1.65 5.5 2.6 4.2 6.5 3.6 2.3 5.5 8 ...
$ BWH        : num  5.5 4.4 1.67 2.95 2.05 1.6 2.15 3.1 1.65 1.45 ...
$ BWD        : num  3.8 4.2 3.75 3.1 3.25 3.8 3.3 3.4 3.75 4.33 ...
$ BWA        : num  1.65 1.72 5.5 2.6 3.9 6.25 3.6 2.3 5.75 7.5 ...
$ IWH        : num  5 5.3 1.67 2.9 2.05 1.63 2.2 3.1 1.63 1.45 ...
$ IWD        : num  3.8 4.2 3.75 3.1 3.1 4 3.25 3.4 3.75 4.4 ...
$ IWA        : num  1.7 1.6 5.3 2.6 4.05 5.5 3.4 2.3 5.7 7.2 ...
$ PSH        : num  5.15 4.73 1.68 2.98 2.1 1.62 2.29 3.13 1.63 1.49 ...
$ PSD        : num  3.84 4.18 3.94 3.14 3.21 3.99 3.31 3.56 3.81 4.34 ...
$ PSA        : num  1.74 1.72 5.47 2.66 4.13 6.13 3.45 2.33 6.38 7.58 ...
$ WHH        : num  5 5.25 1.67 2.9 2.05 1.6 2.25 3 1.62 1.47 ...
$ WHD        : num  3.8 4.2 3.8 3.1 3.2 3.9 3.3 3.5 3.75 4.2 ...
$ WHA        : num  1.7 1.6 5.25 2.62 4 5.8 3.3 2.3 6 8 ...
$ VCH        : num  5 4.75 1.67 2.9 2.1 1.65 2.25 3 1.62 1.45 ...
$ VCD        : num  3.8 4.2 3.9 3.13 3.2 4 3.3 3.5 3.8 4.2 ...
$ VCA        : num  1.75 1.73 5.75 2.7 4.1 5.75 3.3 2.3 5.75 8 ...
$ MaxH       : num  5.5 5.3 1.72 3.05 2.1 1.65 2.31 3.2 1.67 1.52 ...
$ MaxD       : num  3.95 4.4 3.98 3.2 3.3 4.15 3.4 3.56 3.9 4.5 ...
$ MaxA       : num  1.76 1.73 5.75 2.7 4.25 6.5 3.6 2.4 6.5 8.5 ...
$ AvgH       : num  5.07 4.67 1.68 2.91 2.06 1.61 2.23 3.08 1.64 1.47 ...
$ AvgD       : num  3.81 4.12 3.8 3.09 3.18 3.95 3.25 3.41 3.76 4.23 ...
$ AvgA       : num  1.71 1.69 5.29 2.62 4.02 5.8 3.43 2.33 5.78 7.63 ...
$ B365.2.5   : num  1.8 1.53 2 2.3 2.5 1.8 2.1 1.9 2.1 2.2 ...
$ B365.2.5.1 : num  2 2.5 1.8 1.61 1.53 2 1.72 1.9 1.72 1.66 ...
$ P.2.5      : num  1.81 1.52 2.08 2.45 2.72 1.88 2.16 1.95 2.16 2.3 ...
$ P.2.5.1    : num  2.09 2.66 1.82 1.6 1.5 2.02 1.76 1.95 1.76 1.68 ...
$ Max.2.5    : num  1.85 1.53 2.14 2.47 2.75 1.9 2.2 1.98 2.21 2.3 ...
$ Max.2.5.1  : num  2.11 2.72 1.83 1.65 1.54 2.05 1.77 1.95 1.78 1.71 ...
$ Avg.2.5    : num  1.79 1.49 2.07 2.34 2.59 1.84 2.13 1.92 2.13 2.23 ...
$ Avg.2.5.1  : num  2.05 2.58 1.77 1.6 1.49 1.98 1.72 1.89 1.72 1.66 ...
$ AHh        : num  0.75 0.75 -0.75 0 -0.5 -1 -0.25 0.25 -0.75 -1 ...
$ B365AHH    : num  1.99 2.04 1.91 2.05 2.08 2.05 1.95 1.88 1.86 1.88 ...
$ B365AHA    : num  1.94 1.89 2.02 1.88 1.85 1.75 1.98 2.05 2.07 2.05 ...
$ PAHH       : num  1.98 2.01 1.91 2.07 2.1 2.11 1.96 1.9 1.84 1.88 ...
$ PAHA       : num  1.94 1.91 2.01 1.85 1.82 1.81 1.96 2.02 2.08 2.04 ...
$ MaxAHH     : num  2 2.05 1.93 2.07 2.1 2.14 1.97 1.9 1.87 1.89 ...
$ MaxAHA     : num  1.95 1.91 2.03 1.88 1.85 1.85 1.99 2.06 2.08 2.08 ...
```

```
 $ AvgAHH     : num  1.96 2 1.89 2.04 2.06 2.07 1.93 1.87 1.83 1.85 ...
 $ AvgAHA     : num  1.92 1.88 1.99 1.85 1.83 1.8 1.95 2.01 2.06 2.03 ...
 $ B365CH     : num  5.25 5.25 1.66 2.87 1.9 1.53 2.3 3 1.8 1.5 ...
 $ B365CD     : num  3.8 4.2 3.75 3.2 3.1 4 3.4 3.4 3.6 4 ...
 $ B365CA     : num  1.65 1.57 5.5 2.55 5 6.5 3.2 2.4 4.75 8 ...
 $ BWCH       : num  4.75 4.5 1.65 2.95 1.95 1.57 2.35 3 1.8 1.5 ...
 $ BWCD       : num  3.75 4.1 3.8 3.1 3.2 3.8 3.2 3.4 3.4 3.9 ...
 $ BWCA       : num  1.75 1.7 5.5 2.6 4.5 6.5 3.2 2.35 5 7.75 ...
 $ IWCH       : num  5 4.6 1.67 2.9 1.9 1.55 2.35 3 1.85 1.5 ...
 $ IWCD       : num  3.8 3.8 3.8 3.1 3.15 4.05 3.25 3.35 3.55 3.9 ...
 $ IWCA       : num  1.7 1.75 5.3 2.6 4.85 6.3 3.15 2.35 4.4 7.6 ...
 $ PSCH       : num  5.34 5.1 1.69 2.96 1.9 1.54 2.43 3.13 1.82 1.57 ...
 $ PSCD       : num  3.62 4.46 3.88 3.26 3.18 4.19 3.27 3.38 3.53 3.78 ...
 $ PSCA       : num  1.78 1.65 5.47 2.6 5.3 6.87 3.2 2.41 5.07 7.66 ...
 $ WHCH       : num  5 5 1.65 2.9 2.05 1.62 2.25 3 1.78 1.5 ...
 $ WHCD       : num  3.8 4.2 3.9 3.1 3.2 3.9 3.3 3.4 3.5 3.8 ...
 $ WHCA       : num  1.7 1.63 5.25 2.6 4 5.8 3.3 2.35 5 8 ...
 $ VCCH       : num  4.8 5.2 1.7 3 1.9 1.57 2.45 3.13 1.87 1.55 ...
 $ VCCD       : num  3.8 4.4 3.9 3.13 3.2 4 3.3 3.4 3.5 3.9 ...
 $ VCCA       : num  1.8 1.65 5.5 2.63 5.2 7 3.13 2.4 4.6 8 ...
 $ MaxCH      : num  5.8 6 1.72 3.05 1.95 1.58 2.46 3.38 1.87 1.58 ...
 $ MaxCD      : num  3.9 4.52 3.95 3.29 3.26 4.2 3.42 3.47 3.65 4.05 ...
 $ MaxCA      : num  1.81 1.75 6.2 2.72 5.3 7.3 3.58 2.48 5.35 8.9 ...
 $ AvgCH      : num  5.03 4.93 1.68 2.93 1.9 1.54 2.37 3.05 1.83 1.53 ...
 $ AvgCD      : num  3.66 4.26 3.82 3.14 3.16 4.05 3.25 3.34 3.5 3.84 ...
 $ AvgCA      : num  1.76 1.65 5.37 2.59 4.91 6.66 3.18 2.39 4.74 7.68 ...
 $ B365C.2.5  : num  1.9 1.44 2 2.2 2.75 1.9 2.1 2 2 2.37 ...
 $ B365C.2.5.1: num  1.9 2.75 1.8 1.66 1.44 1.9 1.72 1.8 1.8 1.57 ...
 $ PC.2.5     : num  1.98 1.49 2.06 2.2 2.84 1.95 2.18 2.04 2.03 2.43 ...
 $ PC.2.5.1   : num  1.93 2.76 1.85 1.74 1.47 1.95 1.75 1.85 1.87 1.61 ...
 $ MaxC.2.5   : num  1.99 1.51 2.08 2.38 2.85 1.98 2.18 2.09 2.07 2.46 ...
 $ MaxC.2.5.1 : num  2.11 2.88 1.98 1.74 1.5 2.1 1.83 2.05 1.92 1.65 ...
 $ AvgC.2.5   : num  1.86 1.47 2 2.24 2.69 1.9 2.1 1.97 1.99 2.36 ...
 $ AvgC.2.5.1 : num  1.97 2.63 1.82 1.66 1.46 1.92 1.74 1.85 1.83 1.59 ...
 $ AHCh       : num  0.75 1 -0.75 0 -0.5 -1 -0.25 0.25 -0.75 -1 ...
 $ B365CAHH   : num  1.93 1.82 1.94 2.11 1.89 1.96 2.08 1.86 2.02 2.06 ...
 $ B365CAHA   : num  2 1.97 1.99 1.82 2.04 1.97 1.85 2.07 1.77 1.87 ...
  [list output truncated]
```

```
head(d1718); head(d1819); head(d1920)
```

```
  Div     Date   HomeTeam   AwayTeam FTHG FTAG FTR HTHG HTAG HTR HS AS HST AST
1 SP1 18/08/17    Leganes     Alaves    1    0   H    1    0   H 16  6   9   3
2 SP1 18/08/17   Valencia Las Palmas    1    0   H    1    0   H 22  5   6   4
3 SP1 19/08/17      Celta   Sociedad    2    3   A    1    1   D 16 13   5   6
4 SP1 19/08/17     Girona Ath Madrid    2    2   D    2    0   H 13  9   6   3
5 SP1 19/08/17    Sevilla    Espanol    1    1   D    1    1   D  9  9   4   6
6 SP1 20/08/17 Ath Bilbao     Getafe    0    0   D    0    0   D 12  8   2   2
  HF AF HC AC HY AY HR AR B365H B365D B365A  BWH  BWD  BWA  IWH IWD  IWA  LBH
1 14 18  4  2  0  1  0  0  2.05  3.20  4.10 2.05 3.10 4.10 2.10 3.4 3.50 2.05
2 25 13  5  2  3  3  0  1  1.75  3.80  4.50 1.75 3.90 4.60 1.75 3.6 4.80 1.75
3 12 11  5  4  3  1  0  0  2.38  3.25  3.20 2.40 3.30 3.00 2.50 3.3 2.85 2.35
4 15 15  6  0  2  4  0  1  8.00  4.33  1.45 7.50 4.33 1.45 7.20 4.4 1.45 7.50
5 14 12  7  3  2  4  1  0  1.62  4.00  5.50 1.62 3.90 5.75 1.55 4.0 6.20 1.60
```

```
6 16 15  7  6  1  3  0  1  1.50  4.00  7.50 1.48 4.25 7.00 1.50 4.2 6.50 1.50
   LBD  LBA  PSH  PSD  PSA  WHH  WHD  WHA  VCH VCD  VCA Bb1X2 BbMxH BbAvH BbMxD
1 3.00 4.20 2.03 3.25 4.52 2.05 3.10 4.00 2.05 3.2 4.40    35  2.12  2.03  3.40
2 3.80 4.33 1.78 4.01 4.83 1.80 3.75 4.20 1.80 4.0 4.60    35  1.83  1.77  4.04
3 3.25 3.00 2.44 3.40 3.16 2.40 3.40 2.90 2.40 3.4 3.13    35  2.50  2.39  3.50
4 4.00 1.50 8.36 4.38 1.49 8.00 4.20 1.44 7.50 4.3 1.50    35  8.36  7.53  4.40
5 3.90 5.50 1.62 4.17 6.18 1.67 3.60 5.50 1.65 4.0 5.75    35  1.69  1.63  4.17
6 4.00 7.00 1.53 4.37 7.31 1.50 4.00 7.00 1.50 4.2 7.00    34  1.53  1.50  4.40
  BbAvD BbMxA BbAvA BbOU BbMx.2.5 BbAv.2.5 BbMx.2.5.1 BbAv.2.5.1 BbAH BbAHh
1  3.15  4.52  4.17   31     2.84     2.68       1.53       1.46   18 -0.50
2  3.86  4.83  4.46   33     1.69     1.64       2.40       2.27   16 -0.75
3  3.32  3.20  3.01   34     2.03     1.98       1.90       1.84   18 -0.25
4  4.17  1.51  1.48   34     2.20     2.11       1.80       1.74   16  1.25
5  3.93  6.20  5.58   33     1.81     1.75       2.14       2.09   16 -1.00
6  4.17  7.50  6.94   32     2.01     1.94       1.96       1.87   17 -1.00
  BbMxAHH BbAvAHH BbMxAHA BbAvAHA PSCH PSCD PSCA
1    2.07    2.03    1.90    1.86 1.98 3.35 4.63
2    2.05    1.97    1.96    1.91 1.78 4.24 4.43
3    2.08    2.05    1.87    1.83 2.12 3.53 3.74
4    1.77    1.75    2.25    2.16 6.93 3.83 1.63
5    2.12    2.06    1.86    1.82 1.64 4.18 5.82
6    1.90    1.86    2.05    2.01 1.53 4.48 6.91


  Div       Date   HomeTeam   AwayTeam FTHG FTAG FTR HTHG HTAG HTR HS AS HST
1 SP1 17/08/2018      Betis    Levante    0    3   A    0    1   A 22  6   8
2 SP1 17/08/2018     Girona Valladolid    0    0   D    0    0   D 13  2   1
3 SP1 18/08/2018  Barcelona     Alaves    3    0   H    0    0   D 25  3   9
4 SP1 18/08/2018      Celta    Espanol    1    1   D    0    1   A 12 14   2
5 SP1 18/08/2018 Villarreal   Sociedad    1    2   A    1    1   D 16  8   7
6 SP1 19/08/2018      Eibar     Huesca    1    2   A    0    2   A 18  8   6
  AST HF AF HC AC HY AY HR AR B365H B365D B365A  BWH  BWD   BWA  IWH  IWD  IWA
1   4 10 10  5  3  0  2  0  0  1.66  4.00   5.0 1.70  3.7  5.25 1.75 3.60  4.9
2   1 21 20  3  2  1  1  0  0  1.75  3.60   5.0 1.75  3.5  5.25 1.80 3.60  4.5
3   0  6 13  7  1  0  2  0  0  1.11 10.00  21.0 1.11 10.0 20.00 1.12 9.00 20.0
4   5 13 14  8  7  3  2  0  0  1.85  3.50   4.5 1.91  3.4  4.25 1.90 3.50  4.1
5   4 16 10  4  6  2  3  0  0  2.04  3.40   3.8 2.05  3.3  3.90 2.00 3.40  3.8
6   6 12 13  7  0  1  1  0  0  1.66  3.75   5.5 1.70  3.7  5.25 1.70 3.75  5.0
   PSH   PSD   PSA  WHH WHD   WHA  VCH  VCD  VCA Bb1X2 BbMxH BbAvH BbMxD BbAvD
1 1.69  4.19  5.11 1.67 3.9  4.75 1.67  4.2  5.2    40  1.75  1.68  4.25  4.00
2 1.80  3.70  4.99 1.75 3.6  4.60 1.80  3.7  4.8    40  1.85  1.78  3.83  3.60
3 1.11 11.27 25.40 1.08 9.0 29.00 1.10 10.5 34.0    40  1.13  1.10 11.50  9.82
4 1.93  3.64  4.27 1.91 3.5  4.00 1.93  3.5  4.4    38  1.97  1.90  3.73  3.53
5 2.06  3.51  3.91 2.05 3.3  3.60 2.05  3.5  3.9    40  2.11  2.03  3.62  3.43
6 1.72  3.90  5.26 1.73 3.6  4.75 1.70  3.8  5.0    40  1.76  1.70  3.93  3.77
  BbMxA BbAvA BbOU BbMx.2.5 BbAv.2.5 BbMx.2.5.1 BbAv.2.5.1 BbAH BbAHh BbMxAHH
1  5.25  4.95   38     1.82     1.76       2.15       2.06   20 -0.75    1.89
2  5.27  4.79   38     2.21     2.13       1.78       1.71   20 -0.75    2.06
3 41.00 25.67   32     1.39     1.34       3.40       3.18   19 -2.50    1.95
4  4.50  4.20   36     2.13     2.06       1.84       1.76   18 -0.75    2.26
5  3.93  3.76   37     2.05     1.99       1.88       1.81   18 -0.25    1.76
6  5.50  5.08   37     1.95     1.88       1.98       1.91   19 -0.75    1.96
  BbAvAHH BbMxAHA BbAvAHA PSCH  PSCD  PSCA
1    1.85    2.07    2.00 1.59  4.42  5.89
2    2.01    1.90    1.85 1.76  3.57  5.62
```

```
3    1.91    2.00    1.95 1.10 11.85 32.17
4    2.18    1.74    1.71 2.18  3.26  3.85
5    1.74    2.23    2.14 2.32  3.21  3.53
6    1.91    2.01    1.94 1.77  3.68  5.32
```

```
  Div       Date  Time   HomeTeam    AwayTeam FTHG FTAG FTR HTHG HTAG HTR HS AS
1 SP1 16/08/2019 20:00 Ath Bilbao   Barcelona    1    0   H    0    0   D 11 11
2 SP1 17/08/2019 16:00      Celta Real Madrid    1    3   A    0    1   A  7 17
3 SP1 17/08/2019 18:00   Valencia    Sociedad    1    1   D    0    0   D 14 12
4 SP1 17/08/2019 19:00   Mallorca       Eibar    2    1   H    1    0   H 16 11
5 SP1 17/08/2019 20:00    Leganes     Osasuna    0    1   A    0    0   D 13  4
6 SP1 17/08/2019 20:00 Villarreal     Granada    4    4   D    1    1   D 12 14
  HST AST HF AF HC AC HY AY HR AR B365H B365D B365A  BWH  BWD  BWA  IWH  IWD
1   5   2 14  9  3  8  1  1  0  0  5.25  3.80  1.65 5.50 3.80 1.65 5.00 3.80
2   4  11 17 12  6  4  5  2  0  1  4.75  4.20  1.65 4.40 4.20 1.72 5.30 4.20
3   6   3 13 14  3  3  4  4  1  0  1.66  3.75  5.50 1.67 3.75 5.50 1.67 3.75
4   4   5 13 14  9  3  2  3  0  0  2.80  3.20  2.60 2.95 3.10 2.60 2.90 3.10
5   2   2 17 11  8  0  1  4  1  0  2.00  3.20  4.20 2.05 3.25 3.90 2.05 3.10
6   7   7 10 16  2  7  3  1  0  0  1.60  3.80  6.50 1.60 3.80 6.25 1.63 4.00
   IWA  PSH  PSD  PSA  WHH WHD  WHA  VCH  VCD  VCA MaxH MaxD MaxA AvgH AvgD
1 1.70 5.15 3.84 1.74 5.00 3.8 1.70 5.00 3.80 1.75 5.50 3.95 1.76 5.07 3.81
2 1.60 4.73 4.18 1.72 5.25 4.2 1.60 4.75 4.20 1.73 5.30 4.40 1.73 4.67 4.12
3 5.30 1.68 3.94 5.47 1.67 3.8 5.25 1.67 3.90 5.75 1.72 3.98 5.75 1.68 3.80
4 2.60 2.98 3.14 2.66 2.90 3.1 2.62 2.90 3.13 2.70 3.05 3.20 2.70 2.91 3.09
5 4.05 2.10 3.21 4.13 2.05 3.2 4.00 2.10 3.20 4.10 2.10 3.30 4.25 2.06 3.18
6 5.50 1.62 3.99 6.13 1.60 3.9 5.80 1.65 4.00 5.75 1.65 4.15 6.50 1.61 3.95
  AvgA B365.2.5 B365.2.5.1 P.2.5 P.2.5.1 Max.2.5 Max.2.5.1 Avg.2.5 Avg.2.5.1
1 1.71     1.80       2.00  1.81    2.09    1.85      2.11    1.79      2.05
2 1.69     1.53       2.50  1.52    2.66    1.53      2.72    1.49      2.58
3 5.29     2.00       1.80  2.08    1.82    2.14      1.83    2.07      1.77
4 2.62     2.30       1.61  2.45    1.60    2.47      1.65    2.34      1.60
5 4.02     2.50       1.53  2.72    1.50    2.75      1.54    2.59      1.49
6 5.80     1.80       2.00  1.88    2.02    1.90      2.05    1.84      1.98
    AHh B365AHH B365AHA PAHH PAHA MaxAHH MaxAHA AvgAHH AvgAHA B365CH B365CD
1  0.75    1.99    1.94 1.98 1.94   2.00   1.95   1.96   1.92   5.25   3.80
2  0.75    2.04    1.89 2.01 1.91   2.05   1.91   2.00   1.88   5.25   4.20
3 -0.75    1.91    2.02 1.91 2.01   1.93   2.03   1.89   1.99   1.66   3.75
4  0.00    2.05    1.88 2.07 1.85   2.07   1.88   2.04   1.85   2.87   3.20
5 -0.50    2.08    1.85 2.10 1.82   2.10   1.85   2.06   1.83   1.90   3.10
6 -1.00    2.05    1.75 2.11 1.81   2.14   1.85   2.07   1.80   1.53   4.00
  B365CA BWCH BWCD BWCA IWCH IWCD IWCA PSCH PSCD PSCA WHCH WHCD WHCA VCCH VCCD
1   1.65 4.75 3.75 1.75 5.00 3.80 1.70 5.34 3.62 1.78 5.00  3.8 1.70 4.80 3.80
2   1.57 4.50 4.10 1.70 4.60 3.80 1.75 5.10 4.46 1.65 5.00  4.2 1.63 5.20 4.40
3   5.50 1.65 3.80 5.50 1.67 3.80 5.30 1.69 3.88 5.47 1.65  3.9 5.25 1.70 3.90
4   2.55 2.95 3.10 2.60 2.90 3.10 2.60 2.96 3.26 2.60 2.90  3.1 2.60 3.00 3.13
5   5.00 1.95 3.20 4.50 1.90 3.15 4.85 1.90 3.18 5.30 2.05  3.2 4.00 1.90 3.20
6   6.50 1.57 3.80 6.50 1.55 4.05 6.30 1.54 4.19 6.87 1.62  3.9 5.80 1.57 4.00
  VCCA MaxCH MaxCD MaxCA AvgCH AvgCD AvgCA B365C.2.5 B365C.2.5.1 PC.2.5
1 1.80  5.80  3.90  1.81  5.03  3.66  1.76      1.90        1.90   1.98
2 1.65  6.00  4.52  1.75  4.93  4.26  1.65      1.44        2.75   1.49
3 5.50  1.72  3.95  6.20  1.68  3.82  5.37      2.00        1.80   2.06
4 2.63  3.05  3.29  2.72  2.93  3.14  2.59      2.20        1.66   2.20
5 5.20  1.95  3.26  5.30  1.90  3.16  4.91      2.75        1.44   2.84
6 7.00  1.58  4.20  7.30  1.54  4.05  6.66      1.90        1.90   1.95
```

```
  PC.2.5.1 MaxC.2.5 MaxC.2.5.1 AvgC.2.5 AvgC.2.5.1  AHCh B365CAHH B365CAHA
1     1.93     1.99       2.11     1.86       1.97  0.75     1.93     2.00
2     2.76     1.51       2.88     1.47       2.63  1.00     1.82     1.97
3     1.85     2.08       1.98     2.00       1.82 -0.75     1.94     1.99
4     1.74     2.38       1.74     2.24       1.66  0.00     2.11     1.82
5     1.47     2.85       1.50     2.69       1.46 -0.50     1.89     2.04
6     1.95     1.98       2.10     1.90       1.92 -1.00     1.96     1.97
  PCAHH PCAHA MaxCAHH MaxCAHA AvgCAHH AvgCAHA
1  1.91  2.01    2.02    2.03    1.91    1.98
2  1.85  2.07    2.00    2.20    1.82    2.06
3  1.92  2.00    1.96    2.12    1.89    2.00
4  2.09  1.83    2.12    1.88    2.07    1.83
5  1.90  2.01    1.95    2.06    1.90    1.99
6  1.96  1.96    1.98    2.12    1.93    1.95
```

```
View(d1718); View(d1819); View(d1920)
summary(d1718); summary(d1819); summary(d1920)
```

```
     Div                Date              HomeTeam           AwayTeam
 Length:380         Length:380         Length:380         Length:380
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
      FTHG            FTAG            FTR                HTHG
 Min.   :0.000   Min.   :0.000   Length:380         Min.   :0.0000
 1st Qu.:0.750   1st Qu.:0.000   Class :character   1st Qu.:0.0000
 Median :1.000   Median :1.000   Mode  :character   Median :0.0000
 Mean   :1.547   Mean   :1.147                      Mean   :0.6605
 3rd Qu.:2.000   3rd Qu.:2.000                      3rd Qu.:1.0000
 Max.   :7.000   Max.   :6.000                      Max.   :5.0000

      HTAG            HTR                 HS              AS
 Min.   :0.0000   Length:380         Min.   : 2.00   Min.   : 1.00
 1st Qu.:0.0000   Class :character   1st Qu.:10.00   1st Qu.: 8.00
 Median :0.0000   Mode  :character   Median :13.00   Median :10.00
 Mean   :0.4868                      Mean   :13.53   Mean   :10.47
 3rd Qu.:1.0000                      3rd Qu.:16.00   3rd Qu.:13.00
 Max.   :3.0000                      Max.   :30.00   Max.   :24.00

      HST              AST              HF              AF
 Min.   : 0.000   Min.   : 0.000   Min.   : 4.00   Min.   : 0.00
 1st Qu.: 3.000   1st Qu.: 2.000   1st Qu.:11.00   1st Qu.:11.00
 Median : 4.500   Median : 3.000   Median :13.00   Median :14.00
 Mean   : 4.758   Mean   : 3.805   Mean   :13.73   Mean   :13.95
 3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.:17.00   3rd Qu.:17.00
 Max.   :14.000   Max.   :13.000   Max.   :29.00   Max.   :29.00

       HC               AC              HY              AY
 Min.   : 0.000   Min.   : 0.000   Min.   :0.000   Min.   :0.000
 1st Qu.: 4.000   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.:2.000
 Median : 5.000   Median : 4.000   Median :2.000   Median :3.000
```

```
Mean   : 5.613   Mean   : 4.192   Mean   :2.339   Mean   :2.676
3rd Qu.: 7.000   3rd Qu.: 6.000   3rd Qu.:3.000   3rd Qu.:4.000
Max.   :16.000   Max.   :14.000   Max.   :8.000   Max.   :9.000

      HR               AR              B365H            B365D
Min.   :0.0000   Min.   :0.00000   Min.   : 1.050   Min.   : 2.790
1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.: 1.617   1st Qu.: 3.290
Median :0.0000   Median :0.00000   Median : 2.075   Median : 3.500
Mean   :0.1105   Mean   :0.07895   Mean   : 2.777   Mean   : 4.259
3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.: 2.790   3rd Qu.: 4.330
Max.   :2.0000   Max.   :2.00000   Max.   :17.000   Max.   :15.000

    B365A            BWH              BWD              BWA
Min.   : 1.170   Min.   : 1.050   Min.   : 2.950   Min.   : 1.180
1st Qu.: 2.600   1st Qu.: 1.650   1st Qu.: 3.300   1st Qu.: 2.600
Median : 3.700   Median : 2.100   Median : 3.600   Median : 3.700
Mean   : 5.192   Mean   : 2.744   Mean   : 4.278   Mean   : 5.204
3rd Qu.: 5.500   3rd Qu.: 2.750   3rd Qu.: 4.330   3rd Qu.: 5.500
Max.   :34.000   Max.   :14.500   Max.   :15.500   Max.   :34.000

     IWH              IWD              IWA              LBH
Min.   : 1.070   Min.   : 3.050   Min.   : 1.200   Min.   : 1.050
1st Qu.: 1.650   1st Qu.: 3.300   1st Qu.: 2.600   1st Qu.: 1.610
Median : 2.100   Median : 3.500   Median : 3.500   Median : 2.050
Mean   : 2.721   Mean   : 4.161   Mean   : 5.041   Mean   : 2.742
3rd Qu.: 2.700   3rd Qu.: 4.200   3rd Qu.: 5.300   3rd Qu.: 2.750
Max.   :15.000   Max.   :12.000   Max.   :27.000   Max.   :19.000
                                                   NA's   :1
     LBD              LBA              PSH              PSD
Min.   : 2.900   Min.   : 1.170   Min.   : 1.050   Min.   : 3.020
1st Qu.: 3.250   1st Qu.: 2.575   1st Qu.: 1.660   1st Qu.: 3.410
Median : 3.500   Median : 3.600   Median : 2.120   Median : 3.705
Mean   : 4.152   Mean   : 5.375   Mean   : 2.857   Mean   : 4.539
3rd Qu.: 4.200   3rd Qu.: 5.500   3rd Qu.: 2.850   3rd Qu.: 4.455
Max.   :17.000   Max.   :41.000   Max.   :19.650   Max.   :20.380
NA's   :1        NA's   :1
     PSA              WHH              WHD              WHA
Min.   : 1.180   Min.   : 1.060   Min.   : 2.900   Min.   : 1.170
1st Qu.: 2.670   1st Qu.: 1.665   1st Qu.: 3.250   1st Qu.: 2.600
Median : 3.845   Median : 2.100   Median : 3.500   Median : 3.550
Mean   : 5.522   Mean   : 2.738   Mean   : 4.092   Mean   : 5.041
3rd Qu.: 5.942   3rd Qu.: 2.750   3rd Qu.: 4.200   3rd Qu.: 5.500
Max.   :36.500   Max.   :17.000   Max.   :15.000   Max.   :26.000

     VCH              VCD              VCA             Bb1X2
Min.   : 1.040   Min.   : 3.000   Min.   : 1.180   Min.   : 3.00
1st Qu.: 1.650   1st Qu.: 3.400   1st Qu.: 2.630   1st Qu.:35.00
Median : 2.100   Median : 3.700   Median : 3.700   Median :37.00
Mean   : 2.762   Mean   : 4.416   Mean   : 5.472   Mean   :37.71
3rd Qu.: 2.800   3rd Qu.: 4.400   3rd Qu.: 5.750   3rd Qu.:40.00
Max.   :15.000   Max.   :17.000   Max.   :36.000   Max.   :43.00

    BbMxH            BbAvH            BbMxD            BbAvD
Min.   : 1.080   Min.   : 1.050   Min.   : 3.110   Min.   : 2.940
```

```
1st Qu.: 1.700   1st Qu.: 1.640   1st Qu.: 3.478   1st Qu.: 3.328
Median : 2.200   Median : 2.090   Median : 3.750   Median : 3.570
Mean   : 2.966   Mean   : 2.743   Mean   : 4.636   Mean   : 4.261
3rd Qu.: 2.882   3rd Qu.: 2.765   3rd Qu.: 4.553   3rd Qu.: 4.272
Max.   :19.650   Max.   :16.300   Max.   :20.380   Max.   :15.320

    BbMxA            BbAvA            BbOU          BbMx.2.5
Min.   : 1.210   Min.   : 1.170   Min.   : 3.00   Min.   :1.130
1st Qu.: 2.728   1st Qu.: 2.607   1st Qu.:31.75   1st Qu.:1.667
Median : 3.920   Median : 3.665   Median :34.00   Median :1.960
Mean   : 6.107   Mean   : 5.190   Mean   :34.06   Mean   :1.950
3rd Qu.: 6.105   3rd Qu.: 5.543   3rd Qu.:37.00   3rd Qu.:2.203
Max.   :67.000   Max.   :33.420   Max.   :42.00   Max.   :3.080

   BbAv.2.5       BbMx.2.5.1      BbAv.2.5.1         BbAH
Min.   :1.120   Min.   :1.470   Min.   :1.410   Min.   : 1.00
1st Qu.:1.617   1st Qu.:1.780   1st Qu.:1.718   1st Qu.:17.00
Median :1.880   Median :2.000   Median :1.920   Median :18.00
Mean   :1.872   Mean   :2.284   Mean   :2.162   Mean   :18.16
3rd Qu.:2.120   3rd Qu.:2.402   3rd Qu.:2.283   3rd Qu.:19.00
Max.   :2.850   Max.   :7.000   Max.   :5.970   Max.   :24.00

    BbAHh           BbMxAHH         BbAvAHH         BbMxAHA
Min.   :-3.2500   Min.   :1.610   Min.   :1.580   Min.   :1.680
1st Qu.:-0.7500   1st Qu.:1.890   1st Qu.:1.840   1st Qu.:1.897
Median :-0.2500   Median :1.985   Median :1.930   Median :1.970
Mean   :-0.4059   Mean   :1.988   Mean   :1.938   Mean   :1.988
3rd Qu.: 0.0625   3rd Qu.:2.070   3rd Qu.:2.020   3rd Qu.:2.080
Max.   : 2.0000   Max.   :2.420   Max.   :2.340   Max.   :2.520

   BbAvAHA           PSCH            PSCD            PSCA
Min.   :1.630   Min.   : 1.060   Min.   : 2.930   Min.   : 1.160
1st Qu.:1.850   1st Qu.: 1.640   1st Qu.: 3.410   1st Qu.: 2.590
Median :1.930   Median : 2.120   Median : 3.700   Median : 3.850
Mean   :1.937   Mean   : 2.839   Mean   : 4.508   Mean   : 5.695
3rd Qu.:2.030   3rd Qu.: 2.980   3rd Qu.: 4.560   3rd Qu.: 6.095
Max.   :2.440   Max.   :18.700   Max.   :18.500   Max.   :46.000
                NA's   :1        NA's   :1        NA's   :1

    Div               Date             HomeTeam          AwayTeam
Length:380         Length:380         Length:380         Length:380
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character



     FTHG            FTAG           FTR                HTHG
Min.   :0.000   Min.   :0.000   Length:380         Min.   :0.0000
1st Qu.:1.000   1st Qu.:0.000   Class :character   1st Qu.:0.0000
Median :1.000   Median :1.000   Mode  :character   Median :0.0000
Mean   :1.453   Mean   :1.134                      Mean   :0.5447
3rd Qu.:2.000   3rd Qu.:2.000                      3rd Qu.:1.0000
Max.   :8.000   Max.   :6.000                      Max.   :3.0000
     HTAG            HTR                 HS              AS
```

```
Min.   :0.0000   Length:380         Min.   : 3.00   Min.   : 2.00
1st Qu.:0.0000   Class :character   1st Qu.:10.00   1st Qu.: 8.00
Median :0.0000   Mode  :character   Median :13.00   Median :10.00
Mean   :0.5132                     Mean   :13.87   Mean   :10.43
3rd Qu.:1.0000                     3rd Qu.:17.00   3rd Qu.:13.00
Max.   :5.0000                     Max.   :34.00   Max.   :21.00
     HST              AST              HF              AF
Min.   : 0.000   Min.   : 0.000   Min.   : 1.00   Min.   : 3.00
1st Qu.: 3.000   1st Qu.: 2.000   1st Qu.:11.00   1st Qu.:11.00
Median : 5.000   Median : 3.000   Median :13.00   Median :13.00
Mean   : 4.834   Mean   : 3.589   Mean   :13.63   Mean   :13.45
3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.:16.00   3rd Qu.:16.00
Max.   :15.000   Max.   :11.000   Max.   :26.00   Max.   :27.00
      HC               AC               HY              AY
Min.   : 0.000   Min.   : 0.000   Min.   :0.000   Min.   :0.000
1st Qu.: 4.000   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.:2.000
Median : 5.000   Median : 4.000   Median :2.000   Median :3.000
Mean   : 5.574   Mean   : 4.021   Mean   :2.529   Mean   :2.642
3rd Qu.: 7.000   3rd Qu.: 6.000   3rd Qu.:4.000   3rd Qu.:3.000
Max.   :15.000   Max.   :12.000   Max.   :8.000   Max.   :7.000
      HR               AR             B365H           B365D
Min.   :0.00000   Min.   :0.0000   Min.   : 1.080   Min.   : 2.870
1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.: 1.660   1st Qu.: 3.300
Median :0.00000   Median :0.0000   Median : 2.120   Median : 3.500
Mean   :0.08684   Mean   :0.1211   Mean   : 2.596   Mean   : 3.996
3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.: 2.800   3rd Qu.: 4.000
Max.   :1.00000   Max.   :2.0000   Max.   :17.000   Max.   :11.000
    B365A            BWH              BWD              BWA
Min.   : 1.160   Min.   : 1.060   Min.   : 2.900   Min.   : 1.190
1st Qu.: 2.547   1st Qu.: 1.670   1st Qu.: 3.300   1st Qu.: 2.600
Median : 3.500   Median : 2.150   Median : 3.500   Median : 3.500
Mean   : 4.790   Mean   : 2.579   Mean   : 3.991   Mean   : 4.745
3rd Qu.: 5.062   3rd Qu.: 2.800   3rd Qu.: 4.000   3rd Qu.: 5.250
Max.   :29.000   Max.   :15.000   Max.   :12.000   Max.   :36.000
     IWH              IWD              IWA              PSH
Min.   : 1.070   Min.   : 2.850   Min.   : 1.200   Min.   : 1.080
1st Qu.: 1.700   1st Qu.: 3.300   1st Qu.: 2.600   1st Qu.: 1.700
Median : 2.150   Median : 3.500   Median : 3.450   Median : 2.180
Mean   : 2.553   Mean   : 3.943   Mean   : 4.587   Mean   : 2.639
3rd Qu.: 2.763   3rd Qu.: 4.000   3rd Qu.: 4.950   3rd Qu.: 2.840
Max.   :13.000   Max.   :13.000   Max.   :28.000   Max.   :19.070
     PSD              PSA              WHH              WHD
Min.   : 2.990   Min.   : 1.180   Min.   : 1.050   Min.   : 2.500
1st Qu.: 3.357   1st Qu.: 2.652   1st Qu.: 1.700   1st Qu.: 3.300
Median : 3.640   Median : 3.575   Median : 2.150   Median : 3.500
Mean   : 4.133   Mean   : 4.994   Mean   : 2.564   Mean   : 3.997
3rd Qu.: 4.170   3rd Qu.: 5.230   3rd Qu.: 2.800   3rd Qu.: 4.000
Max.   :13.220   Max.   :36.830   Max.   :17.000   Max.   :13.000
     WHA              VCH              VCD              VCA
Min.   : 1.150   Min.   : 1.060   Min.   : 3.000   Min.   : 1.18
1st Qu.: 2.587   1st Qu.: 1.700   1st Qu.: 3.300   1st Qu.: 2.60
Median : 3.500   Median : 2.150   Median : 3.600   Median : 3.60
Mean   : 4.779   Mean   : 2.627   Mean   : 4.097   Mean   : 5.00
3rd Qu.: 5.000   3rd Qu.: 2.800   3rd Qu.: 4.200   3rd Qu.: 5.20
```

```
Max.   :34.000   Max.   :21.000   Max.   :13.000   Max.   :41.00
    Bb1X2            BbMxH            BbAvH            BbMxD
Min.   :31.00   Min.   : 1.100   Min.   : 1.070   Min.   : 3.040
1st Qu.:34.00   1st Qu.: 1.750   1st Qu.: 1.690   1st Qu.: 3.420
Median :36.00   Median : 2.250   Median : 2.160   Median : 3.735
Mean   :36.13   Mean   : 2.739   Mean   : 2.582   Mean   : 4.251
3rd Qu.:38.00   3rd Qu.: 2.913   3rd Qu.: 2.792   3rd Qu.: 4.250
Max.   :41.00   Max.   :21.000   Max.   :16.550   Max.   :15.000
    BbAvD            BbMxA            BbAvA            BbOU
Min.   : 2.920   Min.   : 1.200   Min.   : 1.170   Min.   :28.00
1st Qu.: 3.288   1st Qu.: 2.750   1st Qu.: 2.620   1st Qu.:32.00
Median : 3.550   Median : 3.695   Median : 3.495   Median :34.00
Mean   : 4.005   Mean   : 5.361   Mean   : 4.763   Mean   :33.84
3rd Qu.: 4.032   3rd Qu.: 5.370   3rd Qu.: 5.043   3rd Qu.:35.00
Max.   :12.430   Max.   :52.000   Max.   :33.380   Max.   :39.00
   BbMx.2.5         BbAv.2.5         BbMx.2.5.1       BbAv.2.5.1
Min.   :1.200   Min.   :1.170   Min.   :1.420   Min.   :1.380
1st Qu.:1.710   1st Qu.:1.650   1st Qu.:1.710   1st Qu.:1.650
Median :2.000   Median :1.940   Median :1.950   Median :1.870
Mean   :2.029   Mean   :1.947   Mean   :2.116   Mean   :2.018
3rd Qu.:2.315   3rd Qu.:2.230   3rd Qu.:2.303   3rd Qu.:2.210
Max.   :3.200   Max.   :2.890   Max.   :5.250   Max.   :4.740
     BbAH             BbAHh            BbMxAHH          BbAvAHH
Min.   :15.00   Min.   :-3.0000   Min.   :1.560   Min.   :1.520
1st Qu.:19.00   1st Qu.:-1.0000   1st Qu.:1.850   1st Qu.:1.800
Median :20.00   Median :-0.2500   Median :2.030   Median :1.970
Mean   :19.88   Mean   :-0.4033   Mean   :2.055   Mean   :1.992
3rd Qu.:21.00   3rd Qu.: 0.2500   3rd Qu.:2.200   3rd Qu.:2.130
Max.   :24.00   Max.   : 2.0000   Max.   :3.270   Max.   :3.020
   BbMxAHA          BbAvAHA            PSCH             PSCD
Min.   :1.450   Min.   :1.410   Min.   : 1.070   Min.   : 2.860
1st Qu.:1.800   1st Qu.:1.750   1st Qu.: 1.698   1st Qu.: 3.310
Median :1.950   Median :1.890   Median : 2.190   Median : 3.610
Mean   :1.972   Mean   :1.915   Mean   : 2.726   Mean   : 4.102
3rd Qu.:2.130   3rd Qu.:2.070   3rd Qu.: 2.970   3rd Qu.: 4.202
Max.   :2.850   Max.   :2.670   Max.   :18.040   Max.   :14.910
     PSCA
Min.   : 1.190
1st Qu.: 2.612
Median : 3.645
Mean   : 5.100
3rd Qu.: 5.575
Max.   :36.030

    Div                Date               Time             HomeTeam
Length:380         Length:380         Length:380         Length:380
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character



  AwayTeam              FTHG            FTAG           FTR
Length:380         Min.   :0.000   Min.   :0.000   Length:380
```

```
Class :character   1st Qu.:1.000   1st Qu.:0.000   Class :character
Mode  :character   Median :1.000   Median :1.000   Mode  :character
                   Mean   :1.437   Mean   :1.042
                   3rd Qu.:2.000   3rd Qu.:2.000
                   Max.   :6.000   Max.   :5.000

     HTHG             HTAG          HTR                 HS
Min.   :0.0000   Min.   :0.00   Length:380         Min.   : 3.00
1st Qu.:0.0000   1st Qu.:0.00   Class :character   1st Qu.: 9.00
Median :0.0000   Median :0.00   Mode  :character   Median :12.00
Mean   :0.6026   Mean   :0.45                      Mean   :12.46
3rd Qu.:1.0000   3rd Qu.:1.00                      3rd Qu.:15.00
Max.   :4.0000   Max.   :3.00                      Max.   :25.00

      AS              HST              AST              HF
Min.   : 1.00   Min.   : 0.000   Min.   : 0.000   Min.   : 4.00
1st Qu.: 7.00   1st Qu.: 3.000   1st Qu.: 2.000   1st Qu.:11.00
Median :10.00   Median : 4.000   Median : 3.000   Median :13.00
Mean   :10.14   Mean   : 4.337   Mean   : 3.511   Mean   :13.66
3rd Qu.:12.25   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.:16.00
Max.   :24.00   Max.   :17.000   Max.   :12.000   Max.   :28.00

      AF              HC               AC              HY
Min.   : 5.00   Min.   : 0.000   Min.   : 0.000   Min.   :0.000
1st Qu.:11.00   1st Qu.: 3.000   1st Qu.: 2.750   1st Qu.:1.000
Median :13.00   Median : 5.000   Median : 4.000   Median :2.000
Mean   :13.79   Mean   : 5.042   Mean   : 4.195   Mean   :2.547
3rd Qu.:16.00   3rd Qu.: 7.000   3rd Qu.: 6.000   3rd Qu.:4.000
Max.   :30.00   Max.   :14.000   Max.   :12.000   Max.   :7.000

      AY              HR             AR              B365H
Min.   :0.000   Min.   :0.0   Min.   :0.0000   Min.   : 1.120
1st Qu.:2.000   1st Qu.:0.0   1st Qu.:0.0000   1st Qu.: 1.700
Median :2.000   Median :0.0   Median :0.0000   Median : 2.200
Mean   :2.584   Mean   :0.1   Mean   :0.1263   Mean   : 2.595
3rd Qu.:4.000   3rd Qu.:0.0   3rd Qu.:0.0000   3rd Qu.: 2.900
Max.   :8.000   Max.   :2.0   Max.   :2.0000   Max.   :10.000

    B365D           B365A             BWH              BWD
Min.   :2.800   Min.   : 1.280   Min.   : 1.120   Min.   :2.700
1st Qu.:3.200   1st Qu.: 2.500   1st Qu.: 1.700   1st Qu.:3.200
Median :3.400   Median : 3.500   Median : 2.200   Median :3.400
Mean   :3.805   Mean   : 4.465   Mean   : 2.599   Mean   :3.813
3rd Qu.:4.000   3rd Qu.: 5.062   3rd Qu.: 2.900   3rd Qu.:4.000
Max.   :9.500   Max.   :21.000   Max.   :10.000   Max.   :9.500

     BWA              IWH              IWD              IWA
Min.   : 1.280   Min.   :1.130   Min.   :2.750   Min.   : 1.280
1st Qu.: 2.550   1st Qu.:1.730   1st Qu.:3.188   1st Qu.: 2.550
Median : 3.500   Median :2.200   Median :3.400   Median : 3.450
Mean   : 4.438   Mean   :2.603   Mean   :3.770   Mean   : 4.378
3rd Qu.: 5.250   3rd Qu.:2.913   3rd Qu.:4.000   3rd Qu.: 5.000
Max.   :20.000   Max.   :9.900   Max.   :8.800   Max.   :19.500
```

```
      PSH              PSD              PSA              WHH        
 Min.   : 1.120   Min.   :2.780   Min.   : 1.290   Min.   : 1.110  
 1st Qu.: 1.720   1st Qu.:3.230   1st Qu.: 2.600   1st Qu.: 1.700  
 Median : 2.260   Median :3.515   Median : 3.520   Median : 2.225  
 Mean   : 2.650   Mean   :3.894   Mean   : 4.673   Mean   : 2.616  
 3rd Qu.: 2.958   3rd Qu.:4.088   3rd Qu.: 5.230   3rd Qu.: 2.900  
 Max.   :10.020   Max.   :9.950   Max.   :25.500   Max.   :10.000  
 NA's   :2        NA's   :2       NA's   :2                        
      WHD              WHA              VCH              VCD        
 Min.   :2.800   Min.   : 1.270   Min.   : 1.100   Min.   :2.800  
 1st Qu.:3.200   1st Qu.: 2.550   1st Qu.: 1.700   1st Qu.:3.200  
 Median :3.400   Median : 3.450   Median : 2.200   Median :3.500  
 Mean   :3.802   Mean   : 4.623   Mean   : 2.582   Mean   :3.831  
 3rd Qu.:4.000   3rd Qu.: 5.250   3rd Qu.: 2.885   3rd Qu.:4.025  
 Max.   :9.000   Max.   :26.000   Max.   :10.500   Max.   :9.500  

      VCA              MaxH             MaxD             MaxA       
 Min.   : 1.250   Min.   : 1.160   Min.   : 2.900   Min.   : 1.320  
 1st Qu.: 2.500   1st Qu.: 1.778   1st Qu.: 3.340   1st Qu.: 2.678  
 Median : 3.400   Median : 2.310   Median : 3.600   Median : 3.625  
 Mean   : 4.453   Mean   : 2.757   Mean   : 4.019   Mean   : 4.951  
 3rd Qu.: 5.000   3rd Qu.: 3.055   3rd Qu.: 4.202   3rd Qu.: 5.500  
 Max.   :26.000   Max.   :11.000   Max.   :10.500   Max.   :31.000  

      AvgH             AvgD             AvgA          B365.2.5    
 Min.   :1.120   Min.   :2.780   Min.   : 1.280   Min.   :1.280  
 1st Qu.:1.718   1st Qu.:3.210   1st Qu.: 2.550   1st Qu.:1.800  
 Median :2.220   Median :3.435   Median : 3.455   Median :2.100  
 Mean   :2.608   Mean   :3.813   Mean   : 4.483   Mean   :2.108  
 3rd Qu.:2.917   3rd Qu.:4.022   3rd Qu.: 5.093   3rd Qu.:2.500  
 Max.   :9.910   Max.   :9.160   Max.   :21.610   Max.   :3.400  

   B365.2.5.1        P.2.5          P.2.5.1         Max.2.5     
 Min.   :1.330   Min.   :1.310   Min.   :1.370   Min.   :1.310  
 1st Qu.:1.530   1st Qu.:1.810   1st Qu.:1.570   1st Qu.:1.850  
 Median :1.720   Median :2.145   Median :1.770   Median :2.175  
 Mean   :1.867   Mean   :2.162   Mean   :1.912   Mean   :2.196  
 3rd Qu.:2.000   3rd Qu.:2.520   3rd Qu.:2.098   3rd Qu.:2.553  
 Max.   :3.750   Max.   :3.340   Max.   :3.640   Max.   :3.400  
                 NA's   :2       NA's   :2                      
   Max.2.5.1        Avg.2.5         Avg.2.5.1          AHh        
 Min.   :1.380   Min.   :1.260   Min.   :1.350   Min.   :-2.5000  
 1st Qu.:1.610   1st Qu.:1.780   1st Qu.:1.560   1st Qu.:-0.7500  
 Median :1.820   Median :2.080   Median :1.750   Median :-0.2500  
 Mean   :1.950   Mean   :2.099   Mean   :1.869   Mean   :-0.3289  
 3rd Qu.:2.132   3rd Qu.:2.440   3rd Qu.:2.040   3rd Qu.: 0.0000  
 Max.   :3.950   Max.   :3.160   Max.   :3.680   Max.   : 1.7500  

    B365AHH          B365AHA          PAHH          PAHA       
 Min.   :1.670   Min.   :1.650   Min.   :1.750   Min.   :1.680  
 1st Qu.:1.900   1st Qu.:1.890   1st Qu.:1.890   1st Qu.:1.880  
 Median :1.970   Median :1.960   Median :1.970   Median :1.950  
 Mean   :1.963   Mean   :1.954   Mean   :1.966   Mean   :1.954  
 3rd Qu.:2.040   3rd Qu.:2.030   3rd Qu.:2.040   3rd Qu.:2.030  
```

```
Max.   :2.200   Max.   :2.160   Max.   :2.310   Max.   :2.200
NA's   :10      NA's   :10      NA's   :2       NA's   :2
    MaxAHH          MaxAHA          AvgAHH          AvgAHA
Min.   :1.800   Min.   :1.710   Min.   :1.740   Min.   :1.670
1st Qu.:1.920   1st Qu.:1.910   1st Qu.:1.870   1st Qu.:1.860
Median :1.990   Median :1.980   Median :1.940   Median :1.930
Mean   :1.995   Mean   :1.988   Mean   :1.939   Mean   :1.932
3rd Qu.:2.070   3rd Qu.:2.060   3rd Qu.:2.010   3rd Qu.:2.000
Max.   :2.330   Max.   :2.250   Max.   :2.270   Max.   :2.170

    B365CH          B365CD          B365CA           BWCH
Min.   : 1.100   Min.   : 2.700   Min.   : 1.250   Min.   :1.100
1st Qu.: 1.700   1st Qu.: 3.200   1st Qu.: 2.600   1st Qu.:1.715
Median : 2.150   Median : 3.400   Median : 3.550   Median :2.200
Mean   : 2.598   Mean   : 3.857   Mean   : 4.628   Mean   :2.603
3rd Qu.: 2.870   3rd Qu.: 4.000   3rd Qu.: 5.250   3rd Qu.:2.900
Max.   :11.000   Max.   :10.000   Max.   :26.000   Max.   :9.750

     BWCD            BWCA            IWCH            IWCD
Min.   : 2.750   Min.   : 1.28   Min.   : 1.120   Min.   :2.700
1st Qu.: 3.200   1st Qu.: 2.60   1st Qu.: 1.730   1st Qu.:3.150
Median : 3.400   Median : 3.50   Median : 2.225   Median :3.400
Mean   : 3.817   Mean   : 4.55   Mean   : 2.609   Mean   :3.747
3rd Qu.: 4.000   3rd Qu.: 5.25   3rd Qu.: 2.900   3rd Qu.:4.000
Max.   :10.000   Max.   :23.00   Max.   :11.000   Max.   :9.000

     IWCA            PSCH            PSCD            PSCA
Min.   : 1.250   Min.   : 1.100   Min.   : 2.710   Min.   : 1.270
1st Qu.: 2.600   1st Qu.: 1.720   1st Qu.: 3.188   1st Qu.: 2.688
Median : 3.500   Median : 2.275   Median : 3.480   Median : 3.640
Mean   : 4.368   Mean   : 2.672   Mean   : 3.900   Mean   : 4.873
3rd Qu.: 5.100   3rd Qu.: 2.975   3rd Qu.: 4.062   3rd Qu.: 5.440
Max.   :20.000   Max.   :10.930   Max.   :11.520   Max.   :28.530

     WHCH            WHCD            WHCA            VCCH
Min.   : 1.080   Min.   : 2.620   Min.   : 1.250   Min.   : 1.080
1st Qu.: 1.700   1st Qu.: 3.200   1st Qu.: 2.600   1st Qu.: 1.722
Median : 2.225   Median : 3.400   Median : 3.600   Median : 2.225
Mean   : 2.647   Mean   : 3.827   Mean   : 4.793   Mean   : 2.600
3rd Qu.: 2.900   3rd Qu.: 4.000   3rd Qu.: 5.250   3rd Qu.: 2.880
Max.   :11.000   Max.   :11.000   Max.   :26.000   Max.   :10.500

     VCCD            VCCA            MaxCH           MaxCD
Min.   : 2.750   Min.   : 1.250   Min.   : 1.130   Min.   : 2.860
1st Qu.: 3.200   1st Qu.: 2.600   1st Qu.: 1.788   1st Qu.: 3.320
Median : 3.450   Median : 3.400   Median : 2.355   Median : 3.610
Mean   : 3.845   Mean   : 4.549   Mean   : 2.836   Mean   : 4.062
3rd Qu.: 4.000   3rd Qu.: 5.050   3rd Qu.: 3.105   3rd Qu.: 4.242
Max.   :10.500   Max.   :26.000   Max.   :13.000   Max.   :12.400

    MaxCA           AvgCH           AvgCD           AvgCA
Min.   : 1.320   Min.   : 1.100   Min.   : 2.710   Min.   : 1.280
1st Qu.: 2.743   1st Qu.: 1.710   1st Qu.: 3.167   1st Qu.: 2.618
Median : 3.870   Median : 2.235   Median : 3.430   Median : 3.545
```

```
 Mean   : 5.215   Mean   : 2.625   Mean   : 3.824   Mean   : 4.630
 3rd Qu.: 5.750   3rd Qu.: 2.908   3rd Qu.: 4.032   3rd Qu.: 5.282
 Max.   :31.370   Max.   :10.390   Max.   :10.410   Max.   :24.600

   B365C.2.5      B365C.2.5.1       PC.2.5         PC.2.5.1        MaxC.2.5
 Min.   :1.220   Min.   :1.30   Min.   :1.220   Min.   :1.320   Min.   :1.260
 1st Qu.:1.720   1st Qu.:1.53   1st Qu.:1.795   1st Qu.:1.570   1st Qu.:1.857
 Median :2.100   Median :1.72   Median :2.125   Median :1.790   Median :2.190
 Mean   :2.142   Mean   :1.87   Mean   :2.188   Mean   :1.922   Mean   :2.241
 3rd Qu.:2.500   3rd Qu.:2.10   3rd Qu.:2.530   3rd Qu.:2.110   3rd Qu.:2.560
 Max.   :3.500   Max.   :4.33   Max.   :3.720   Max.   :4.520   Max.   :3.720

   MaxC.2.5.1       AvgC.2.5       AvgC.2.5.1         AHCh
 Min.   :1.330   Min.   :1.220   Min.   :1.290   Min.   :-2.7500
 1st Qu.:1.627   1st Qu.:1.760   1st Qu.:1.550   1st Qu.:-0.7500
 Median :1.840   Median :2.080   Median :1.750   Median :-0.2500
 Mean   :1.997   Mean   :2.118   Mean   :1.878   Mean   :-0.3329
 3rd Qu.:2.223   3rd Qu.:2.440   3rd Qu.:2.062   3rd Qu.: 0.0000
 Max.   :4.610   Max.   :3.520   Max.   :4.130   Max.   : 1.7500

    B365CAHH        B365CAHA         PCAHH          PCAHA
 Min.   :1.700   Min.   :1.670   Min.   :1.750   Min.   :1.750
 1st Qu.:1.880   1st Qu.:1.880   1st Qu.:1.880   1st Qu.:1.880
 Median :1.960   Median :1.960   Median :1.960   Median :1.960
 Mean   :1.959   Mean   :1.956   Mean   :1.962   Mean   :1.959
 3rd Qu.:2.040   3rd Qu.:2.040   3rd Qu.:2.040   3rd Qu.:2.040
 Max.   :2.160   Max.   :2.160   Max.   :2.200   Max.   :2.210

    MaxCAHH         MaxCAHA         AvgCAHH        AvgCAHA
 Min.   :1.780   Min.   :1.800   Min.   :1.72   Min.   :1.750
 1st Qu.:1.940   1st Qu.:1.930   1st Qu.:1.86   1st Qu.:1.860
 Median :2.020   Median :2.020   Median :1.93   Median :1.940
 Mean   :2.021   Mean   :2.016   Mean   :1.94   Mean   :1.938
 3rd Qu.:2.100   3rd Qu.:2.100   3rd Qu.:2.02   3rd Qu.:2.010
 Max.   :2.260   Max.   :2.270   Max.   :2.15   Max.   :2.180
```

```
# Ahora seleccionaremos únicamente las columnas
# 'Date', 'HomeTeam', 'AwayTeam', 'FTHG', 'FTAG'
# y 'FTR' en cada uno de los data frames. Primero
# guardaremos los data frames en una lista con nombre
# 'lista' y después con ayuda de las funciones 'lapply'
# y 'select' (del paquete 'dplyr'), seleccionaremos las
# columnas deseadas. Los nuevos data frames quedarán guardados en 'nlista'.

lista <- list(d1718, d1819, d1920)
nlista <- lapply(lista, select, Date, HomeTeam, AwayTeam, FTHG, FTAG, FTR)

# Con las funciones 'lapply' y 'str' observaremos
# la estrucura de nuestros nuevos data frames

lapply(nlista, str)
```

```
'data.frame':   380 obs. of  6 variables:
```

```
 $ Date    : chr  "18/08/17" "18/08/17" "19/08/17" "19/08/17" ...
 $ HomeTeam: chr  "Leganes" "Valencia" "Celta" "Girona" ...
 $ AwayTeam: chr  "Alaves" "Las Palmas" "Sociedad" "Ath Madrid" ...
 $ FTHG    : int  1 1 2 2 1 0 2 0 1 0 ...
 $ FTAG    : int  0 0 3 2 1 0 0 3 0 1 ...
 $ FTR     : chr  "H" "H" "A" "D" ...
'data.frame':   380 obs. of  6 variables:
 $ Date    : chr  "17/08/2018" "17/08/2018" "18/08/2018" "18/08/2018" ...
 $ HomeTeam: chr  "Betis" "Girona" "Barcelona" "Celta" ...
 $ AwayTeam: chr  "Levante" "Valladolid" "Alaves" "Espanol" ...
 $ FTHG    : int  0 0 3 1 1 1 2 1 2 1 ...
 $ FTAG    : int  3 0 0 1 2 2 0 4 1 1 ...
 $ FTR     : chr  "A" "D" "H" "D" ...
'data.frame':   380 obs. of  6 variables:
 $ Date    : chr  "16/08/2019" "17/08/2019" "17/08/2019" "17/08/2019" ...
 $ HomeTeam: chr  "Ath Bilbao" "Celta" "Valencia" "Mallorca" ...
 $ AwayTeam: chr  "Barcelona" "Real Madrid" "Sociedad" "Eibar" ...
 $ FTHG    : int  1 1 1 2 0 4 1 0 1 1 ...
 $ FTAG    : int  0 3 1 1 1 4 0 2 2 0 ...
 $ FTR     : chr  "H" "A" "D" "H" ...

[[1]]
NULL

[[2]]
NULL

[[3]]
NULL
```

```r
# Arreglamos las columnas 'Date' para que 'R'
# reconozca los elementos como fechas, esto
# lo hacemos con las funciones 'mutate' (paquete 'dplyr') y 'as.Date'.

nlista[[1]] <- mutate(nlista[[1]], Date = as.Date(Date, "%d/%m/%y"))
nlista[[2]] <- mutate(nlista[[2]], Date = as.Date(Date, "%d/%m/%Y"))
nlista[[3]] <- mutate(nlista[[3]], Date = as.Date(Date, "%d/%m/%Y"))

# Verificamos que nuestros cambios se hayan llevado a cabo

lapply(nlista, str)
```

```
'data.frame':   380 obs. of  6 variables:
 $ Date    : Date, format: "2017-08-18" "2017-08-18" ...
 $ HomeTeam: chr  "Leganes" "Valencia" "Celta" "Girona" ...
 $ AwayTeam: chr  "Alaves" "Las Palmas" "Sociedad" "Ath Madrid" ...
 $ FTHG    : int  1 1 2 2 1 0 2 0 1 0 ...
 $ FTAG    : int  0 0 3 2 1 0 0 3 0 1 ...
 $ FTR     : chr  "H" "H" "A" "D" ...
'data.frame':   380 obs. of  6 variables:
 $ Date    : Date, format: "2018-08-17" "2018-08-17" ...
 $ HomeTeam: chr  "Betis" "Girona" "Barcelona" "Celta" ...
 $ AwayTeam: chr  "Levante" "Valladolid" "Alaves" "Espanol" ...
```

```
 $ FTHG    : int  0 0 3 1 1 1 2 1 2 1 ...
 $ FTAG    : int  3 0 0 1 2 2 0 4 1 1 ...
 $ FTR     : chr  "A" "D" "H" "D" ...
'data.frame':   380 obs. of  6 variables:
 $ Date    : Date, format: "2019-08-16" "2019-08-17" ...
 $ HomeTeam: chr  "Ath Bilbao" "Celta" "Valencia" "Mallorca" ...
 $ AwayTeam: chr  "Barcelona" "Real Madrid" "Sociedad" "Eibar" ...
 $ FTHG    : int  1 1 1 2 0 4 1 0 1 1 ...
 $ FTAG    : int  0 3 1 1 1 4 0 2 2 0 ...
 $ FTR     : chr  "H" "A" "D" "H" ...

[[1]]
NULL

[[2]]
NULL

[[3]]
NULL
```

```
# Finalmente, con ayuda de las funciones
# 'rbind' y 'do.call' combinamos los data
# frames contenidos en 'nlista' como un único data frame

data <- do.call(rbind, nlista)
dim(data)
```

```
[1] 1140    6
```

```
str(data)
```

```
'data.frame':   1140 obs. of  6 variables:
 $ Date    : Date, format: "2017-08-18" "2017-08-18" ...
 $ HomeTeam: chr  "Leganes" "Valencia" "Celta" "Girona" ...
 $ AwayTeam: chr  "Alaves" "Las Palmas" "Sociedad" "Ath Madrid" ...
 $ FTHG    : int  1 1 2 2 1 0 2 0 1 0 ...
 $ FTAG    : int  0 0 3 2 1 0 0 3 0 1 ...
 $ FTR     : chr  "H" "H" "A" "D" ...
```

```
tail(data)
```

```
           Date HomeTeam    AwayTeam FTHG FTAG FTR
1135 2020-07-19  Espanol       Celta    0    0   D
1136 2020-07-19  Granada  Ath Bilbao    4    0   H
1137 2020-07-19  Leganes Real Madrid    2    2   D
1138 2020-07-19  Levante      Getafe    1    0   H
1139 2020-07-19  Osasuna    Mallorca    2    2   D
1140 2020-07-19  Sevilla    Valencia    1    0   H
```

```
View(data)
summary(data)
```

```
      Date               HomeTeam           AwayTeam              FTHG
 Min.   :2017-08-18   Length:1140        Length:1140        Min.   :0.000
 1st Qu.:2018-03-17   Class :character   Class :character   1st Qu.:1.000
 Median :2019-01-16   Mode  :character   Mode  :character   Median :1.000
 Mean   :2019-01-15                                         Mean   :1.479
 3rd Qu.:2019-10-27                                         3rd Qu.:2.000
 Max.   :2020-07-19                                         Max.   :8.000
      FTAG           FTR
 Min.   :0.000   Length:1140
 1st Qu.:0.000   Class :character
 Median :1.000   Mode  :character
 Mean   :1.108
 3rd Qu.:2.000
 Max.   :6.000
```

```r
# 1. Primero debemos crear el data frame 'SmallData',
# que contenga las columnas 'date', 'home.team',
# 'home.score', 'away.team' y 'away.score'; esto
# lo hacemos con ayuda de la función 'select' del paquete
# 'dplyr'. Luego establecemos un directorio de trabajo y
# con ayuda de la función 'write.csv' guardamos nuestro
# data frame como un archivo csv con nombre *soccer.csv*.

SmallData <- select(data, date = Date, home.team = HomeTeam,
                    home.score = FTHG, away.team = AwayTeam,
                    away.score = FTAG)
write.csv(x = SmallData, file = "soccer.csv", row.names = FALSE)

# 2. Ahora con la función 'create.fbRanks.dataframes'
# del paquete 'fbRanks' importamos nuestro archivo
# *soccer.csv* a 'R' y al mismo tiempo asignamos a
# una variable llamada 'listasoccer'. Se creará una
# lista con los elementos 'scores' y 'teams' que son
# data frames listos para la función 'rank.teams'.
# Asignamos estos data frames a variables llamadas
# 'anotaciones' y 'equipos'.

listasoccer <- create.fbRanks.dataframes(scores.file = "soccer.csv")
```

```
Alert: teams info file was not passed in.
Will construct one from the scores data frame but teams in the scores file must use a unique name.
Alert: teams resolver was not passed in.
Will construct one from the team info data frame.
```

```r
anotaciones <- listasoccer$scores
equipos <- listasoccer$teams

# 3. Ahora con ayuda de la función 'unique'
# creamos un vector de fechas que no se repiten
# y que corresponde a las fechas en las que se
# jugaron partidos. Creamos una variable llamada
# 'n' que contiene el número de fechas diferentes.
# Posteriormente, con la función 'rank.teams' y usando
```

```r
# como argumentos los data frames 'anotaciones' y 'equipos',
# creamos un ranking de equipos usando unicamente datos desde
# la fecha inicial y hasta la penúltima fecha en la que se
# jugaron partidos, estas fechas las especificamos en 'max.date'
# y 'min.date'. Guardamos los resultados con el nombre 'ranking'.

fecha <- unique(anotaciones$date)
n <- length(fecha)
ranking <- rank.teams(scores = anotaciones, teams = equipos,
                      max.date = fecha[n-1],
                      min.date = fecha[1])
```

```
Team Rankings based on matches 2017-08-18 to 2020-07-16
   team        total attack defense n.games.Var1 n.games.Freq
1  Barcelona    1.51 2.23   1.28    Barcelona    113
2  Ath Madrid   1.24 1.33   1.78    Ath Madrid   113
3  Real Madrid  1.15 1.86   1.19    Real Madrid  113
4  Valencia     0.56 1.34   1.10    Valencia     113
5  Getafe       0.55 1.10   1.33    Getafe       113
6  Sevilla      0.43 1.37   0.98    Sevilla      113
7  Granada      0.37 1.26   1.03    Granada       37
8  Villarreal   0.33 1.38   0.91    Villarreal   113
9  Sociedad     0.32 1.39   0.90    Sociedad     113
10 Ath Bilbao   0.15 1.02   1.09    Ath Bilbao   113
11 Osasuna      0.07 1.18   0.89    Osasuna       37
12 Betis        0.05 1.28   0.81    Betis        113
13 Celta        0.02 1.26   0.81    Celta        113
14 Eibar       -0.02 1.08   0.91    Eibar        113
15 Levante     -0.03 1.26   0.78    Levante      113
16 Girona      -0.18 1.07   0.83    Girona        76
17 Espanol     -0.21 0.93   0.93    Espanol      113
18 Alaves      -0.23 0.95   0.90    Alaves       113
19 Leganes     -0.31 0.82   0.98    Leganes      113
20 Valladolid  -0.33 0.79   1.00    Valladolid    75
21 Huesca      -0.35 1.09   0.72    Huesca        38
22 Mallorca    -0.41 1.02   0.74    Mallorca      37
23 Vallecano   -0.51 1.04   0.67    Vallecano     38
24 La Coruna   -0.82 0.94   0.60    La Coruna     38
25 Malaga      -1.17 0.58   0.76    Malaga        38
26 Las Palmas  -1.43 0.59   0.63    Las Palmas    38
```

```r
# 4. Finalmente estimamos las probabilidades de los eventos,
# el equipo de casa gana, el equipo visitante gana o el resultado
# es un empate para los partidos que se jugaron en la última fecha
# de nuestro vector de fechas 'fecha'. Esto lo hacemos con ayuda de
# la función 'predict' y usando como argumentos 'ranking' y 'fecha[n]'
# que especificamos en 'date'.

pred <- predict(ranking, date = fecha[n])
```

```
Predicted Match Results for 1900-05-01 to 2100-06-01
Model based on data from 2017-08-18 to 2020-07-16
```

```
--------------------------------------------
2020-07-19 Alaves vs Barcelona, HW 9%, AW 75%, T 15%, pred score 0.7-2.5  actual: AW (0-5)
2020-07-19 Valladolid vs Betis, HW 29%, AW 43%, T 28%, pred score 1-1.3  actual: HW (2-0)
2020-07-19 Villarreal vs Eibar, HW 45%, AW 30%, T 25%, pred score 1.5-1.2  actual: HW (4-0)
2020-07-19 Ath Madrid vs Sociedad, HW 54%, AW 20%, T 27%, pred score 1.5-0.8  actual: T (1-1)
2020-07-19 Espanol vs Celta, HW 32%, AW 41%, T 27%, pred score 1.2-1.4  actual: T (0-0)
2020-07-19 Granada vs Ath Bilbao, HW 40%, AW 31%, T 29%, pred score 1.2-1  actual: HW (4-0)
2020-07-19 Leganes vs Real Madrid, HW 13%, AW 66%, T 21%, pred score 0.7-1.9  actual: T (2-2)
2020-07-19 Levante vs Getafe, HW 25%, AW 48%, T 27%, pred score 0.9-1.4  actual: HW (1-0)
2020-07-19 Osasuna vs Mallorca, HW 48%, AW 27%, T 25%, pred score 1.6-1.1  actual: T (2-2)
2020-07-19 Sevilla vs Valencia, HW 34%, AW 40%, T 26%, pred score 1.2-1.4  actual: HW (1-0)
```