Trabajo Práctico Integador

Mª Cristina Heredia Gómez

Análisis de los datos de Regresión

Para regresión, la base de datos que me fue asignada es *ForestFires*, mientras que para clasificación se me asignó el dataset *tae*.

Cálculo de estadísticos y descripción de los datos

```
setwd("/home/cris/mrcrstnherediagmez@gmail.com/Intro_Ciencia_de_datos/Trabajo Final/Datasets Regresion/
forestFires<-read.csv("forestFires.dat", comment.char="@")
str(forestFires)
```

```
## 'data.frame':    516 obs. of  13 variables:
##  $ X1    : int  7 4 8 6 4 4 7 6 2 6 ...
##  $ X3    : int  4 5 6 3 3 4 4 5 4 5 ...
##  $ X9    : int  3 9 3 9 8 8 10 4 9 9 ...
##  $ X7    : int  1 6 7 4 7 6 5 4 6 1 ...
##  $ X91   : num  90.1 92.5 89.3 92.8 81.6 90.2 90 81.5 93.4 90.9 ...
##  $ X276.3: num  39.7 88 51.3 119 56.7 ...
##  $ X825.1: num  86.6 698.6 102.2 783.5 665.6 ...
##  $ X7.1  : num  6.2 7.1 9.6 7.5 1.9 8.9 8.7 2.7 8.1 7 ...
##  $ X21.9 : num  16.1 20.3 11.5 18.9 27.8 18.4 11.3 5.8 28.6 21.3 ...
##  $ X43   : int  29 45 39 34 32 42 60 54 27 42 ...
##  $ X4    : num  3.1 3.1 5.8 7.2 2.7 6.7 5.4 5.8 2.2 2.2 ...
##  $ X0    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ X70.76: num  1.75 0 7.19 34.36 6.44 ...
```

```
summary(forestFires)
```

```
##        X1              X3              X9              X7        
##  Min.   :1.000   Min.   :2.000   Min.   : 1.000   Min.   :1.000  
##  1st Qu.:3.000   1st Qu.:4.000   1st Qu.: 7.000   1st Qu.:2.000  
##  Median :4.000   Median :4.000   Median : 8.000   Median :5.000  
##  Mean   :4.676   Mean   :4.302   Mean   : 7.473   Mean   :4.254  
##  3rd Qu.:7.000   3rd Qu.:5.000   3rd Qu.: 9.000   3rd Qu.:6.000  
##  Max.   :9.000   Max.   :9.000   Max.   :12.000   Max.   :7.000  
##       X91            X276.3           X825.1           X7.1       
##  Min.   :18.70   Min.   :  1.10   Min.   :  7.9   Min.   : 0.000  
##  1st Qu.:90.20   1st Qu.: 67.03   1st Qu.:436.6   1st Qu.: 6.475  
##  Median :91.60   Median :108.30   Median :663.0   Median : 8.400  
##  Mean   :90.64   Mean   :110.55   Mean   :547.4   Mean   : 9.025  
##  3rd Qu.:92.90   3rd Qu.:141.57   3rd Qu.:713.9   3rd Qu.:10.850  
##  Max.   :96.20   Max.   :291.30   Max.   :860.6   Max.   :56.100  
##      X21.9            X43              X4              X0        
##  Min.   : 2.20   Min.   : 15.00   Min.   :0.400   Min.   :0.0000  
##  1st Qu.:15.50   1st Qu.: 32.75   1st Qu.:2.700   1st Qu.:0.0000  
##  Median :19.30   Median : 41.50   Median :4.000   Median :0.0000  
##  Mean   :18.88   Mean   : 44.29   Mean   :4.018   Mean   :0.0217  
##  3rd Qu.:22.80   3rd Qu.: 53.00   3rd Qu.:4.900   3rd Qu.:0.0000
```

```
##  Max.   :33.30   Max.   :100.00   Max.   :9.400   Max.   :6.4000
##      X70.76
##  Min.   :   0.000
##  1st Qu.:   0.000
##  Median :   0.520
##  Mean   :  12.735
##  3rd Qu.:   6.548
##  Max.   :1090.840
```

Comenzamos cargando el dataset *ForestFires*. Con *str* podemos ver que se trata de un dataframe de 516 muestras con 13 variables, todas de tipo numérico. Algunas variables son de tipo entero mientras que otras sí contienen valores con decimales. Usamos *summary* para obtener más información acerca de estas variables, en concreto su mínimo, máximo, media, mediana, 1º y 3º cuartil. Podemos ver que hay algunas variables, como por ejemplo X1, X3, X4 donde el valor de la media y el de la mediana son próximos, mientras que en otras variables, como X70.76 el valor de la mediana es 0.520 frente al de la media que es 12.735. Si a ésto le añadimos que el mínimo de esa variable es 0.000 y que el máximo es 1090.840, ésto podría indicar que en la variable X70.76 existe un sesgo importante hacia números pequeños cercanos al mínimo.

A pesar de que con *summary* tenemos un resumen de algunos estadísticos del dataset, calculamos otros adicionales como la desviación estándar(SD), desviación absoluta media(MAD), asimetría y curtosis y el rango intercuartílico (IQR) del mismo, que nos puedan arrojar más información sobre el dataset al que nos enfrentamos. En primer lugar, comprobamos si hay valores perdidos en el dataset:

```
sum(which(is.na(forestFires)))
```

```
## [1] 0
```

Podemos comprobar que el dataset no contiene ningún valor perdido. A continuación calculamos la media, mediana y desviación típica de cada variable.

```
apply(forestFires,2,mean)
```

```
##           X1           X3           X9           X7          X91
##   4.67635659   4.30232558   7.47286822   4.25387597  90.64399225
##       X276.3       X825.1         X7.1        X21.9          X43
## 110.55174419 547.40290698   9.02538760  18.88333333  44.29069767
##           X4           X0       X70.76
##   4.01763566   0.02170543  12.73505814
```

```
apply(forestFires,2,median)
```

```
##     X1     X3     X9     X7    X91 X276.3 X825.1   X7.1  X21.9    X43
##   4.00   4.00   8.00   5.00  91.60 108.30 663.00   8.40  19.30  41.50
##     X4     X0 X70.76
##   4.00   0.00   0.52
```

```
apply(forestFires,2,sd)
```

```
##          X1          X3          X9          X7         X91      X276.3
##   2.3103614   1.2297582   2.2772067   2.0714154   5.5254454  63.6920356
##      X825.1        X7.1       X21.9         X43          X4          X0
## 248.0057961   4.5631146   5.8107428  16.3332049   1.7933911   0.2962448
##      X70.76
##  63.6663671
```

Podemos ver que hay variables como X276.3, X43, X70.76 y especialmente X825.1 que presentan una desviación típica muy alta, por lo que presentan una alta variabilidad con respecto a su media. Una forma más robusta de medir la desviación que presentan las variables es la MAD (median absolute deviation), que calcula la media de las desviaciones absolutas a un punto central, que por defecto será la mediana:

```r
apply(forestFires,2,mad)
```

```
##         X1         X3         X9         X7        X91     X276.3
##   2.965200   1.482600   1.482600   2.965200   1.927380  51.742740
##     X825.1       X7.1      X21.9        X43         X4         X0
## 116.977140   3.113460   5.337360  15.567300   1.927380   0.000000
##     X70.76
##   0.770952
```

Esta vez X276.3, X825.1 y X43 presentan una variabilidad alta con respecto a su mediana, a diferencia de X70.76 que con respecto a su mediana no presenta tal variabilidad. Otra medida robusta para medir la variabilidad es el rango intercuartil (IQR), que se calcula a partir de los cuartiles 1º y 3º:

```r
apply(forestFires,2,quantile)
```

```
##      X1 X3 X9 X7  X91  X276.3 X825.1   X7.1 X21.9    X43  X4  X0    X70.76
## 0%    1  2  1  1 18.7   1.100    7.9  0.000   2.2  15.00 0.4 0.0    0.0000
## 25%   3  4  7  2 90.2  67.025  436.6  6.475  15.5  32.75 2.7 0.0    0.0000
## 50%   4  4  8  5 91.6 108.300  663.0  8.400  19.3  41.50 4.0 0.0    0.5200
## 75%   7  5  9  6 92.9 141.575  713.9 10.850  22.8  53.00 4.9 0.0    6.5475
## 100%  9  9 12  7 96.2 291.300  860.6 56.100  33.3 100.00 9.4 6.4 1090.8400
```

```r
apply(forestFires,2,IQR)
```

```
##       X1       X3       X9       X7      X91   X276.3   X825.1      X7.1
##   4.0000   1.0000   2.0000   4.0000   2.7000  74.5500 277.3000   4.3750
##    X21.9      X43       X4       X0   X70.76
##   7.3000  20.2500   2.2000   0.0000   6.5475
```

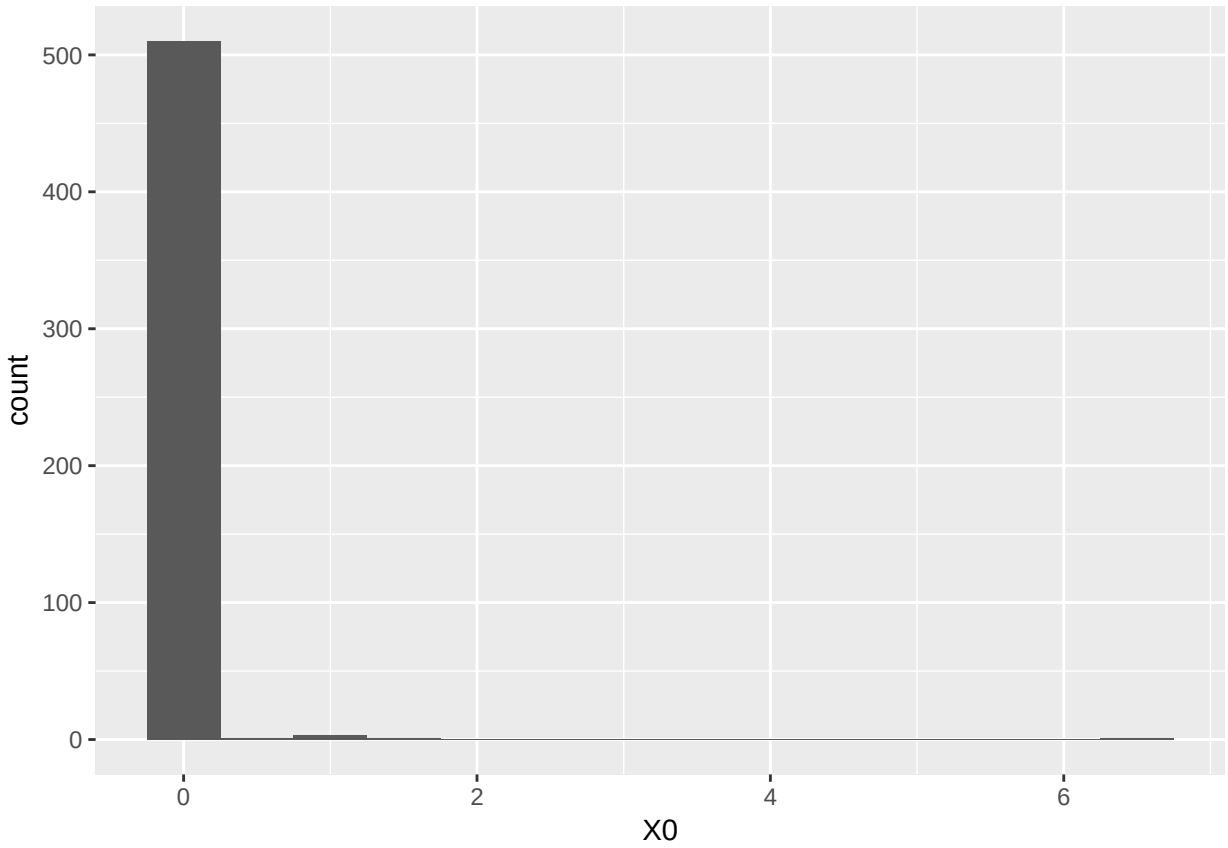
Con *quantile* calculamos los tres cuartiles de cada variable del dataset, y calculamos su IQR. Puesto que el IQR define el rango en el que se encuentran el 50% de los valores, podemos deducir que variables que presentan un IQR muy alto, como por ejemplo X276.3, X825.1 y X43 indican variabilidad en los datos, especialmente X825.1 donde el IQR es de 277.3000 y, por tanto, no está normalmente distribuida. Por último vamos a calcular las medidas de *skewness* y *curtosis*, que nos darán indicativos de como de asimétricas y picudas son las distribuciones de as variables:

```r
library(moments)
skewness(forestFires)
```

```
##          X1          X3          X9          X7         X91      X276.3
##  0.03491498  0.41318183 -1.21157468 -0.21003167 -6.54987376  0.53721187
##      X825.1        X7.1       X21.9         X43          X4          X0
## -1.09639488  2.52551565 -0.32740457  0.85912345  0.56873527 19.73956825
##      X70.76
## 12.83197949
```

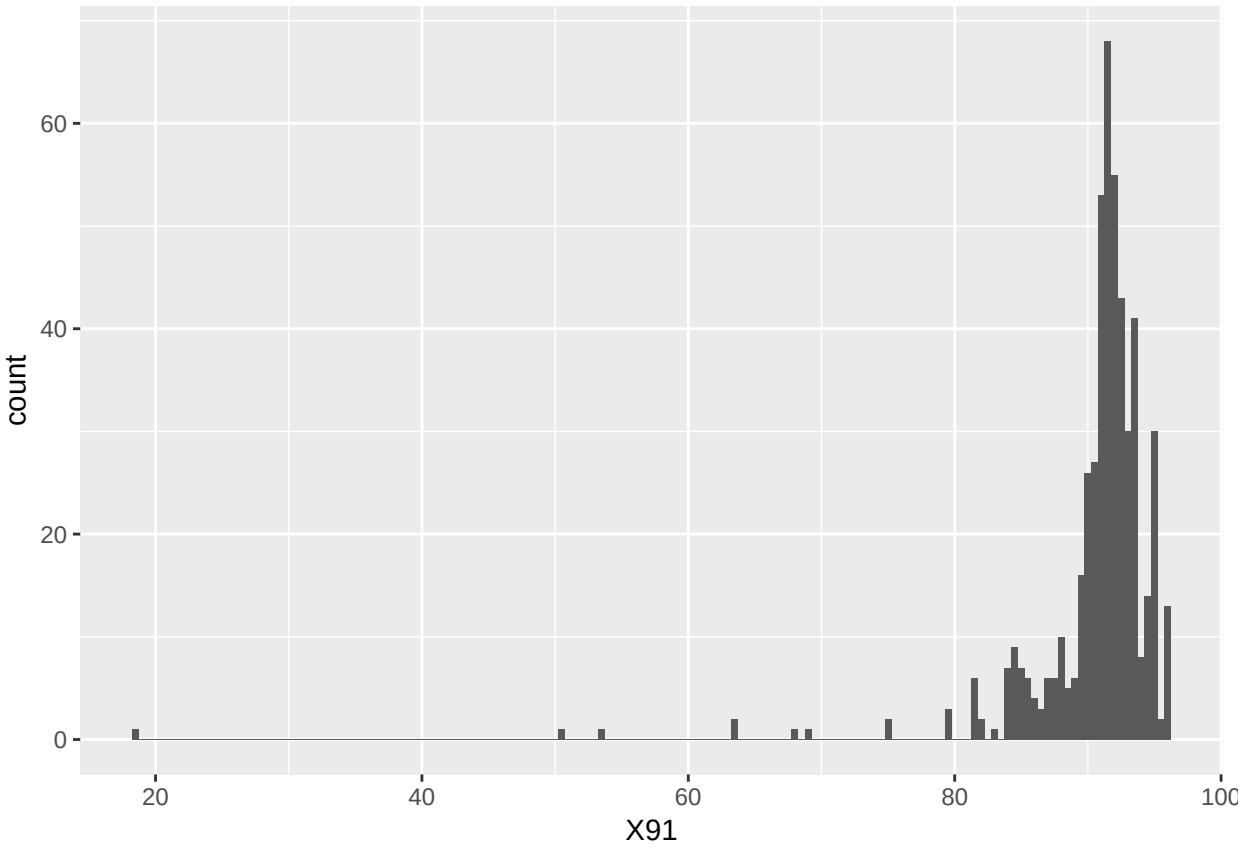
Valores de *skewness* positivos o negativos lejanos a 0 indican un sesgo o desplazamiento en la distribución. Podemos ver que, en este dataset, prácticamente todas las variables presentan algún desplazamiento en su distribución, si bien algunos son especialmente llamativos, como por ejemplo la variable XO que tiene skewness=19.73956825 y por tanto presenta un sesgo positivo, esto es, los datos se concentran a la izquierda de la distribución, quedando una cola más larga a la derecha y corta a la izquierda:

```r
library(ggplot2)
ggplot(forestFires, aes(X0))+geom_histogram(binwidth = 0.5)
```



Algo similar ocurre con la variable X70.06, que presenta también un sesgo positivo. Sin embargo, con la variable X91 ocurre lo inverso, es decir, presenta un sesgo negativo:

```
ggplot(forestFires, aes(X91))+geom_histogram(binwidth = 0.5)
```



Pues, como se observa en el gráfico, los datos se concentran a la derecha, quedando una cola larga a la izquierda.

```
kurtosis(forestFires)
```

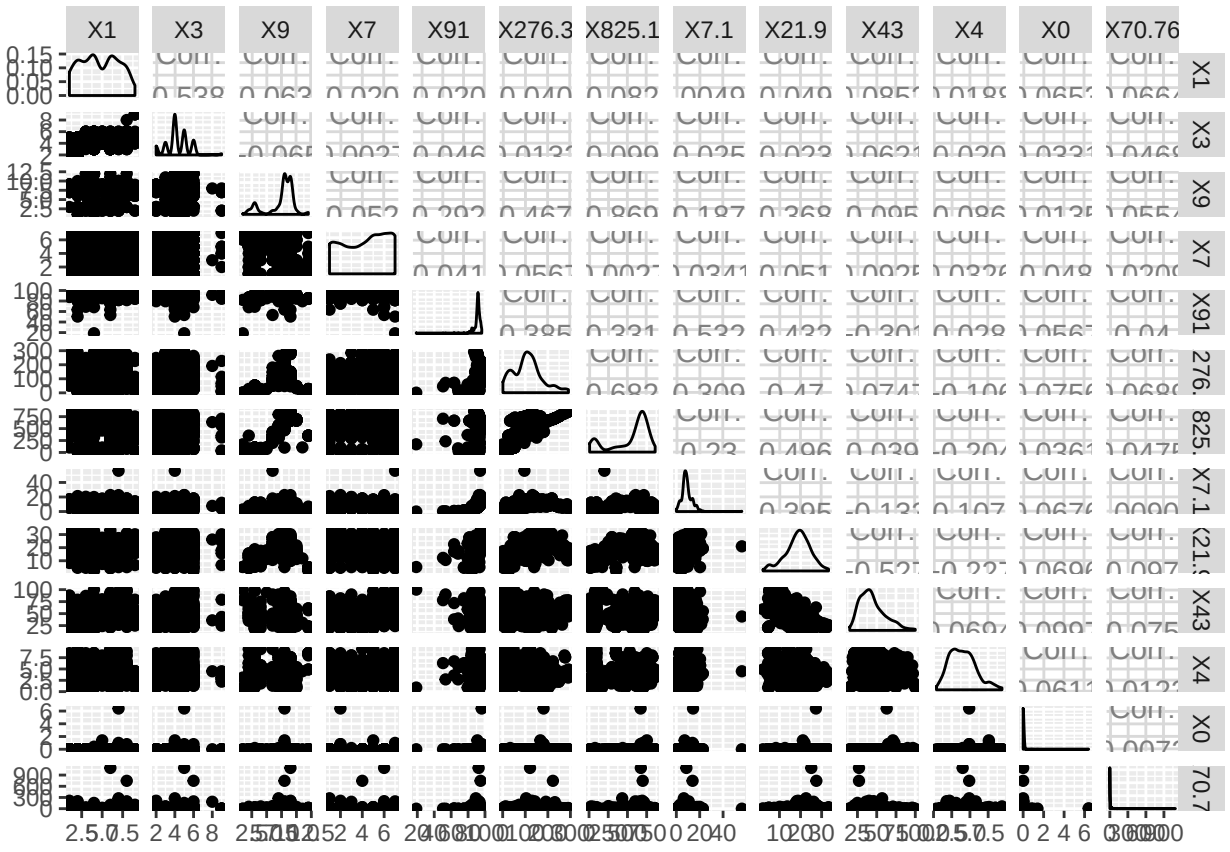
```
##         X1         X3         X9         X7        X91     X276.3
##   1.828938   4.400015   3.619169   1.715468  69.271199   3.191381
##     X825.1       X7.1      X21.9        X43         X4         X0
##   2.740991  24.200985   3.119023   3.415311   3.036290 419.405689
##     X70.76
## 195.596115
```

Por último, los valores de curtosis nos indican como de picuda es la distribución en el centro. Por ejemplo, X0 y X70.76 son especialmente picudas en 0. Está claro que ambas variables no siguen una distribucion normal.

Visualización

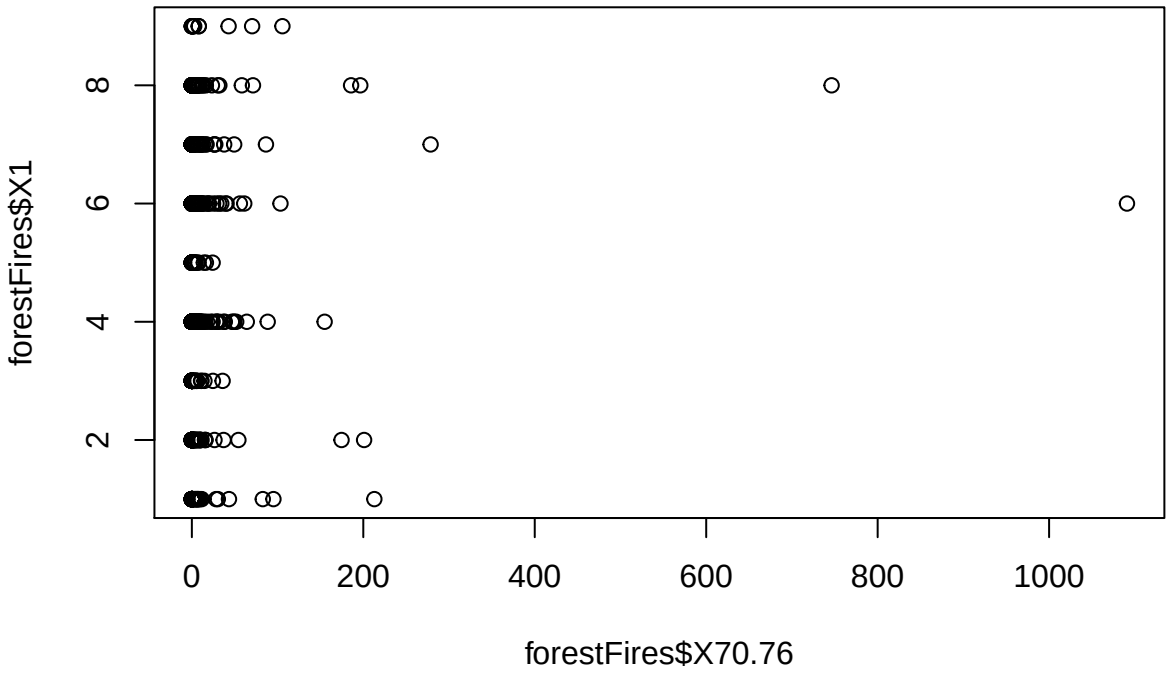
Para entender mejor los datos, hacemos un gráfico con *ggpairs* que nos dará información sobre correlación de las variables y gráficos en pares de las mismas:

```
library("GGally")
ggpairs(data=forestFires)
```



A simple vista en este primer plot, podemos ver que parece que la variable respuesta (X70.76) que representa el área de bosque quemada, tiene una correlación inferior a 0.1 con cualquiera de las otras variables. También podemos verlo en los minigráficos que nos crea, donde parece que el valor de las otras variables no influye en el valor de X70.76. Por ejemplo, representamos X1 frente a X70.76 para ilustrarlo mejor:

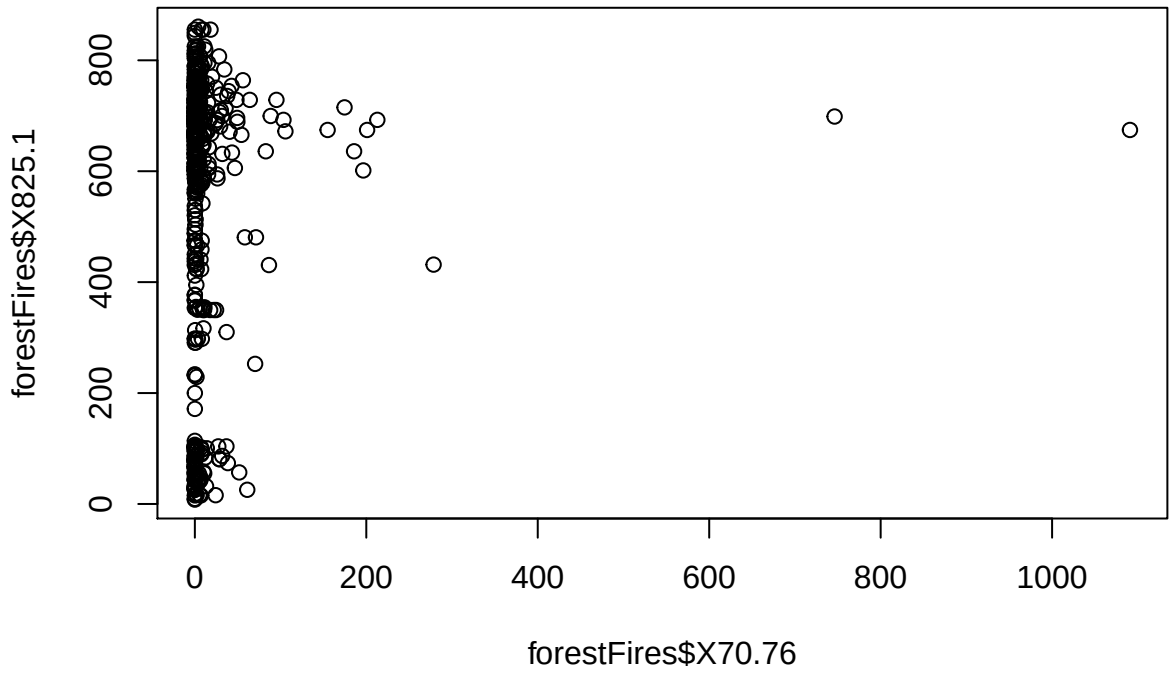
```
plot(forestFires$X70.76,forestFires$X1)
```



donde

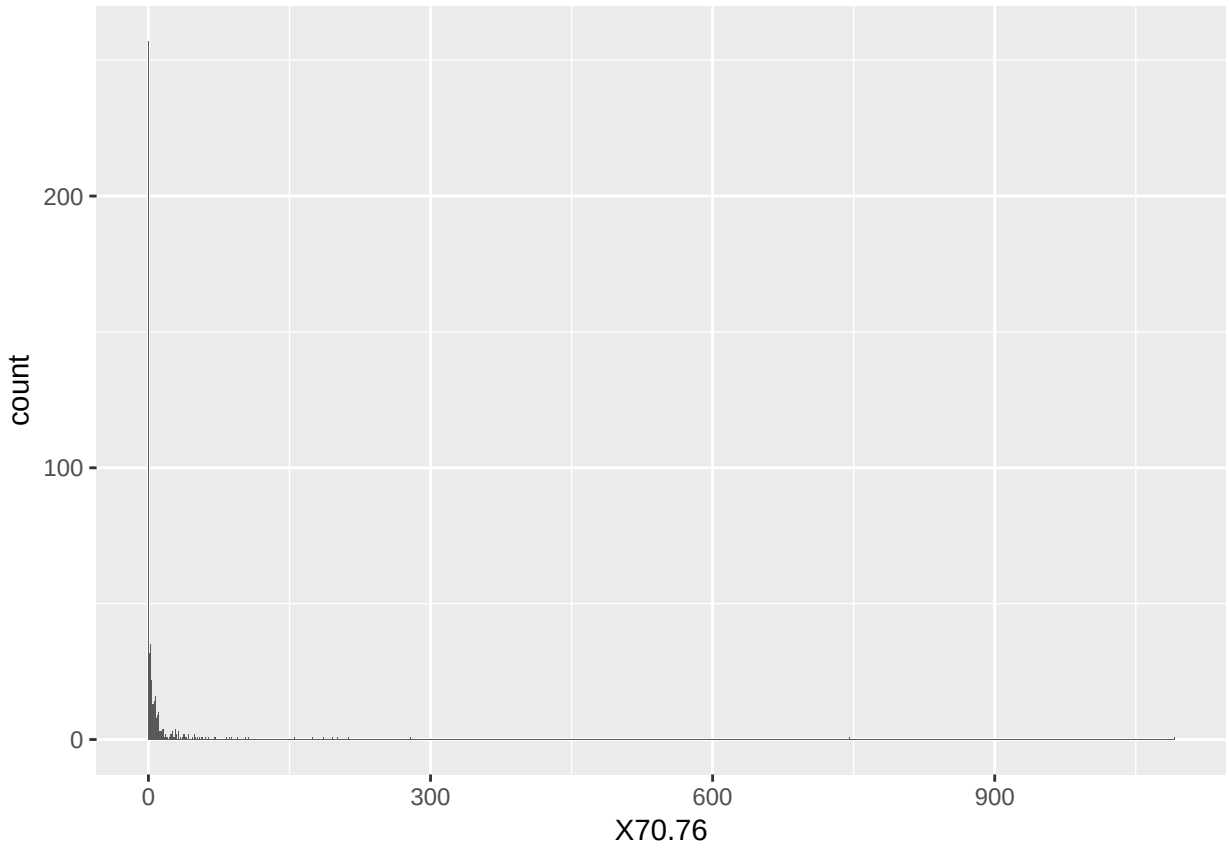
podemos ver que, sin importar si X1 toma valor 0, 2, 6 o 9, el valor del área calcinada es mayoritariamente inferior a 200 hectáreas, a excepción de dos puntos que podrían ser outliers. Lo mismo ocurre con el resto de variables, por ejemplo, x825.1:

```
plot(forestFires$X70.76,forestFires$X825.1)
```



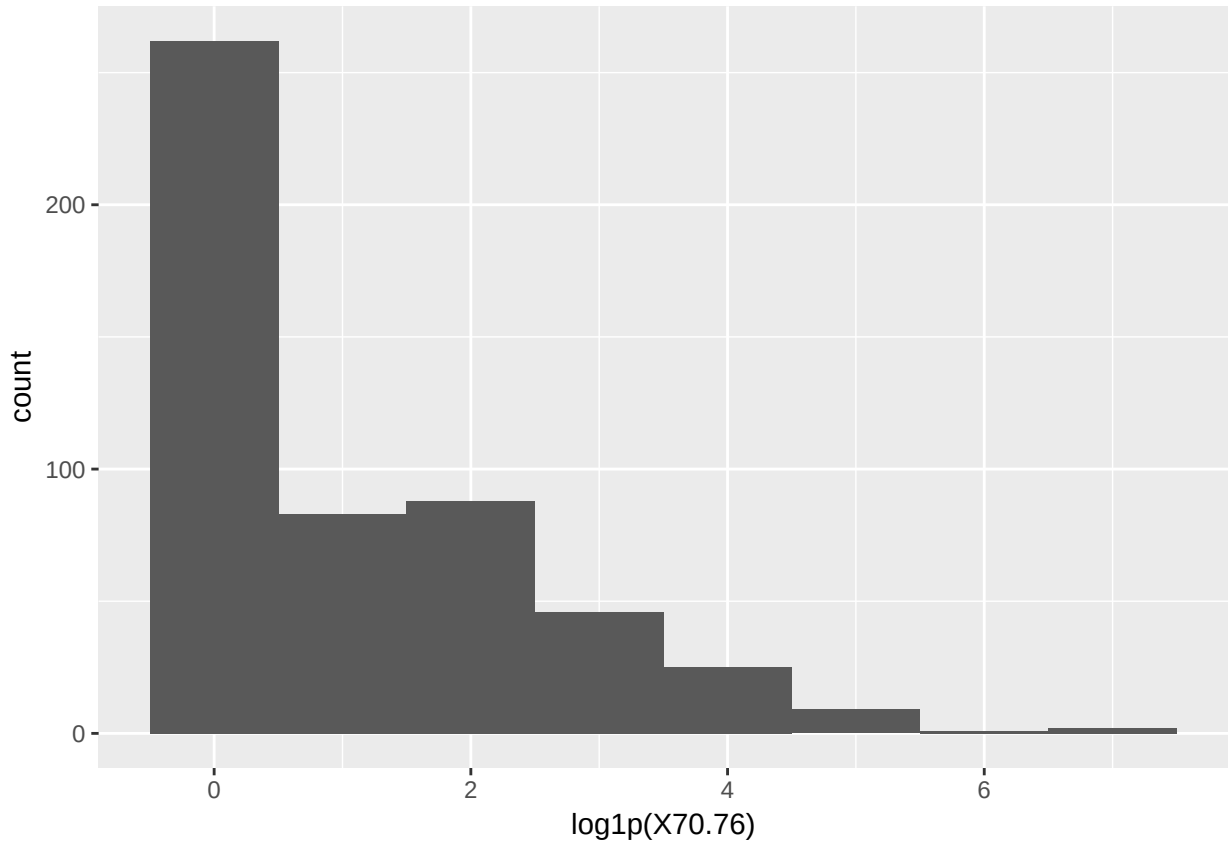
donde, aparentemente, sus valores no influyen significtivamente en el área calcinada. Si analizamos de nuevo la variable salida:

```
ggplot(forestFires, aes(X70.76))+geom_histogram(binwidth = 1)
```



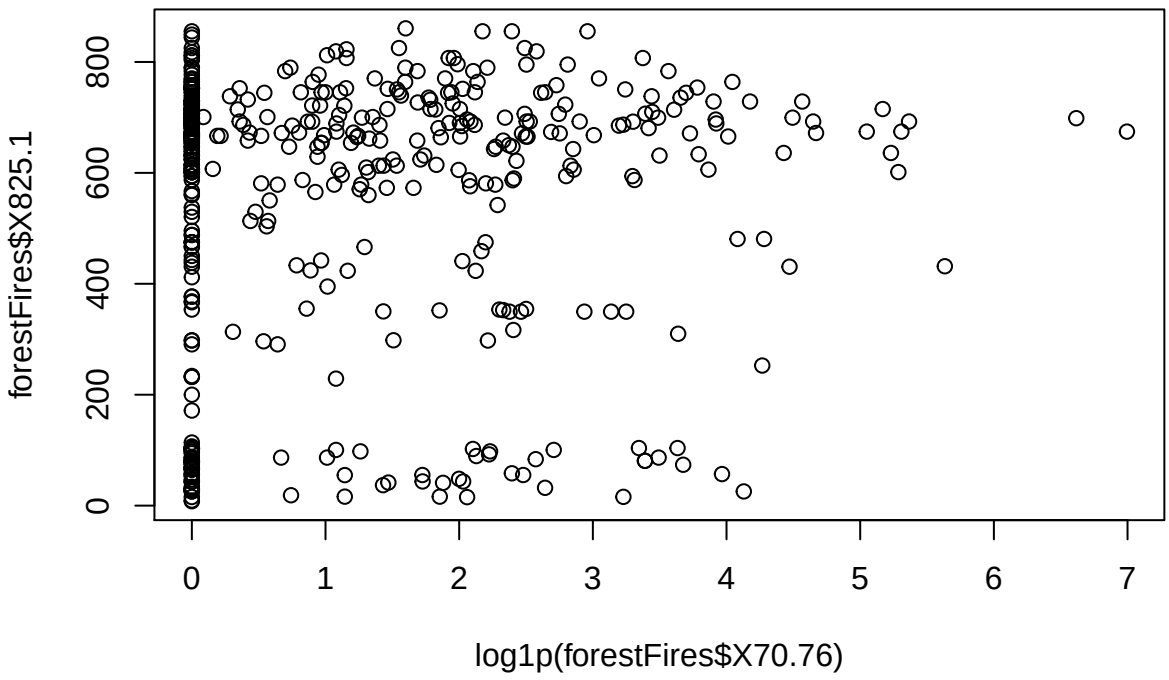
recordemos que esta variable presentaba un sesgo positivo muy grande, dado que la mayoría de sus valores se concentraba a la izquierda de la distribución. En concreto, se concentran mayoritariamente en el valor 0, significando esto que en la mayoría de los incendios recogidos el número de hectáreas quemadas fue 0 (inferior a los 100m2). Sin embargo, los puntos más distantes y menos frecuentes correspondientes a hectáreas quemadas de mayor tamaño parecen no ser outliers, sino casos menos frecuentes intrínsecos en los datos. Probamos a aplicar una tansformación logaritmica que solucione este problema en dicha variable, en concreto *ln(x+1)* tal y como recomiendan los autores del dataset en la descripción del mismo: http://www3.dsi.uminho.pt/pcortez/forestfires/forestfires-names.txt.

```
ggplot(forestFires, aes(log1p(X70.76)))+geom_histogram(binwidth = 1)
```



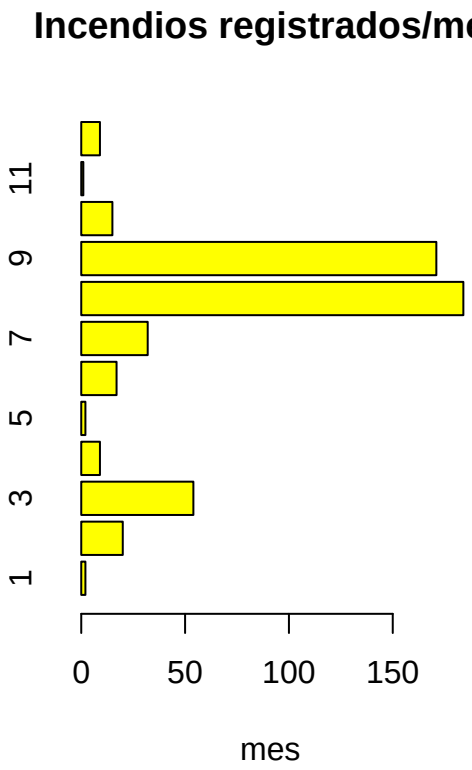
Vemos que con la transformación mejora la dispersión de los datos con respecto a su forma original. Por ejemplo, si ahora la volvemos a representar frente a X825.1:

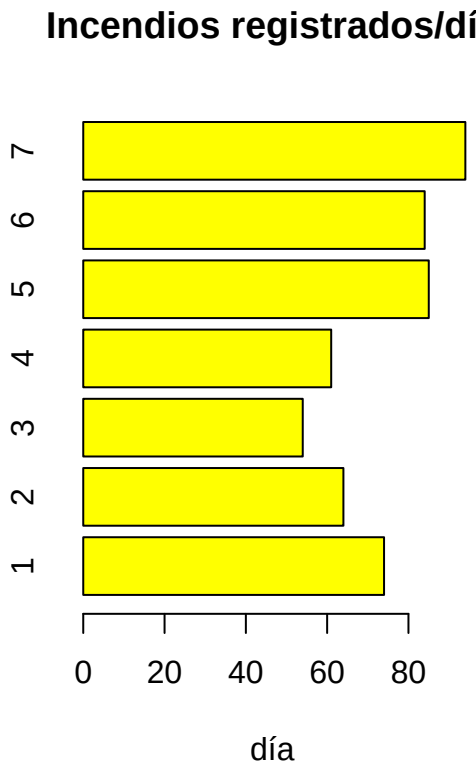
```
plot(log1p(forestFires$X70.76),forestFires$X825.1)
```



Vemos que los datos ahora están más despegados de la izquierda, y aunque no presentan un patrón insituitivo visible, sí se podría intentar utilizarlos para explicar la salida. Con el propósito de entender mejor los datos, podemos ver en qué meses y días de los mismos de distribuyen los incendios:

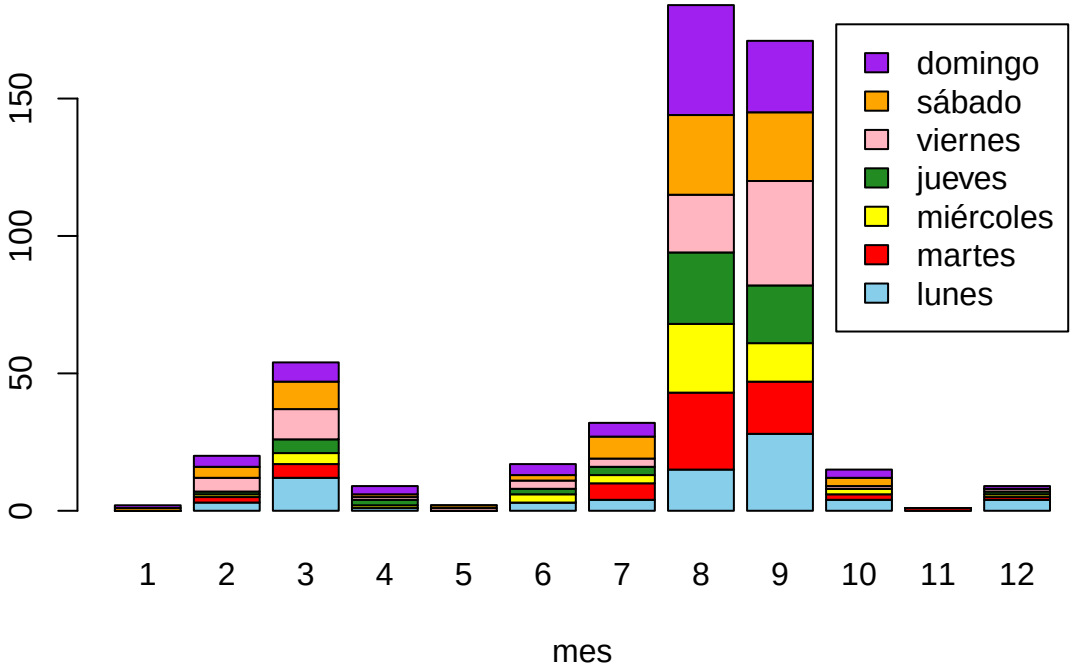
```
par(mfrow=c(1,2))
barplot(table(forestFires$X9),col = "yellow", xlab="mes",horiz=TRUE,main="Incendios registrados/mes")
barplot(table(forestFires$X7),col = "yellow", xlab="día",horiz=TRUE,main="Incendios registrados/día")
```





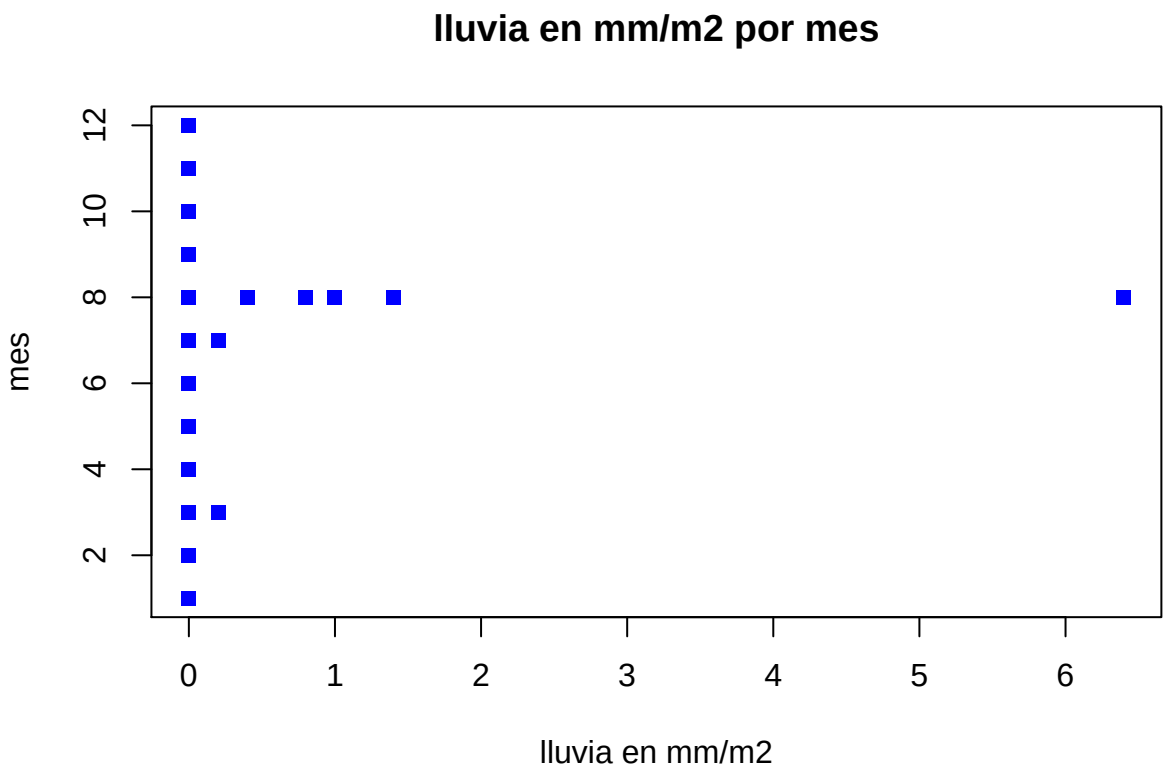
Vemos que el mayor número de incendios se registra en los meses de agosto y septiembre con mucha diferencia del resto de meses. Les siguen los meses de marzo y julio, con 54 y 32 incendios registrados respectivamente, mientras que los meses que menos incendios presentan son los meses de invierno junto con abril y mayo. Con respecto a los días, los días de la semana en de los que más incendios hay registrados son fines de semana, especialmente el Domingo (94), seguido de sábado (84) y viernes (85). Si representamos los días que ha habido incendios frente a los meses:

```
counts<-table(forestFires$X7,forestFires$X9)
barplot(counts,col=c("skyblue","red","yellow","forestgreen","lightpink","orange","purple"),
        legend =c("lunes","martes","miércoles","jueves","viernes","sábado","domingo"), xlab = "mes",mai
```



Podemos ver que en los meses 8 y 9 (agosto y septiembre) que son los meses con más incendios registados, la mayoría de los incendios en estos meses se producen en fines de semana y viernes. Si representamos ahora la lluvia en mm/m2 frente a los meses:

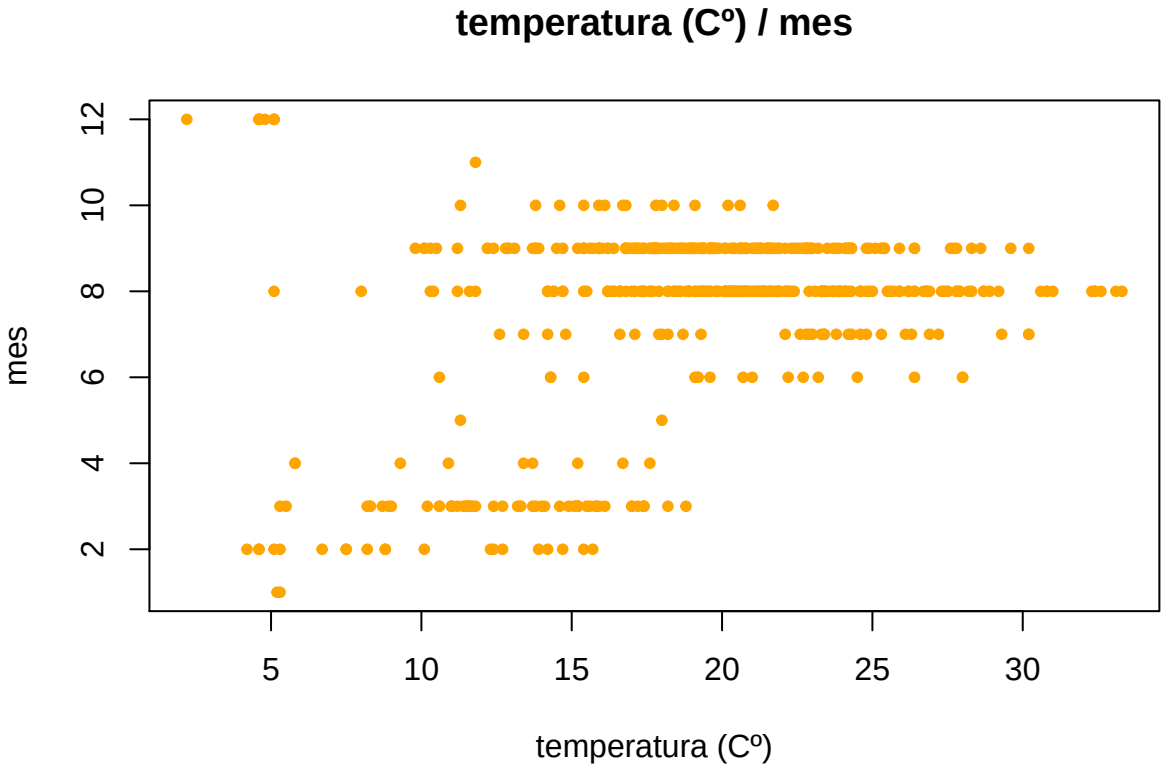
```
plot((forestFires$X0),forestFires$X9,xlab = "lluvia en mm/m2", ylab = "mes",
     main = "lluvia en mm/m2 por mes",pch=15,col="blue")
```



Obeservamos que, en general para cualquier mes cuando se produjeron incendios no llovió, excepto en agosto, que es el mes donde más registros de lluvia se tienen, habiendo un día en el que se registraron 6.4 mm/m2, seguramente por una tormenta de verano.

También podemos representar la temperatura el celsius frente al mes:

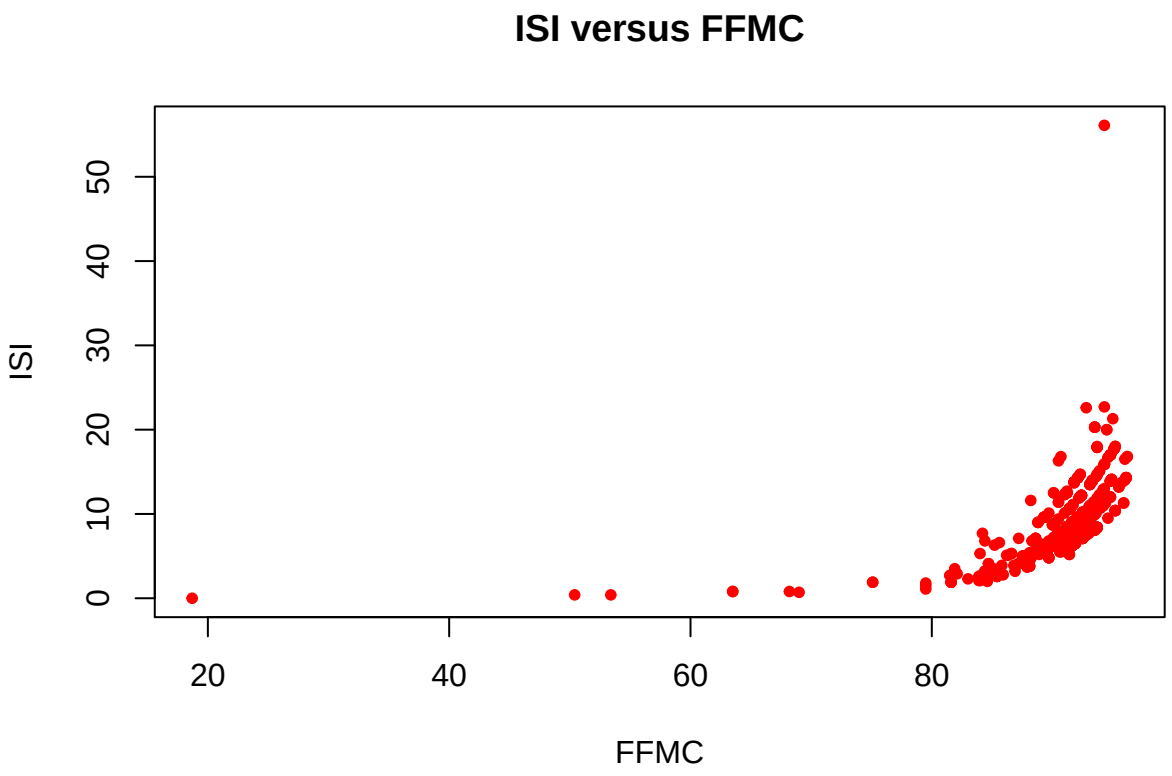
```
plot(forestFires$X21.9,forestFires$X9,pch=20,col="orange",xlab = "temperatura (Cº)",ylab="mes",
     main = "temperatura (Cº) / mes")
```



Parece coherente que los meses de invierno tengan registradas temperaturas más bajas que los meses de verano, así como que los en los meses de verano se registren las temperaturas más altas, por lo tanto, estas variables podrían estar explicadas la una en la otra.

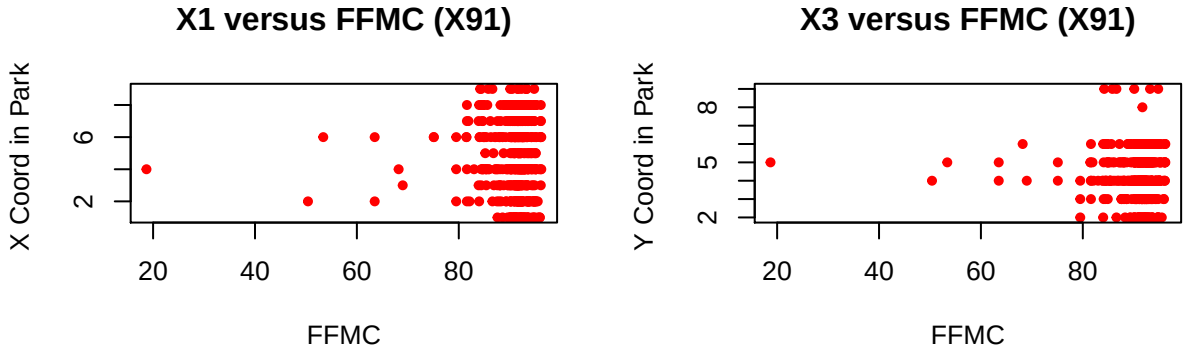
Éste dataset contiene 12 variables (más la variable respuesta). Tras leer lo que representa cada variable en la descripción del dataset que dan los autores, citada más arriba, era lógico pensar que algunas variables guardaran relaciones entre sí, por ejemplo, el FFMC (fuel moisture code) que influye en la rapidez de propagación del fuego y el ISI, una medida relacionada con la velocidad de propagación del fuego. Si los representamos:

```
plot(forestFires$X91,forestFires$X7.1,pch=20,col="red",xlab="FFMC", ylab = "ISI",
     main = "ISI versus FFMC")
```

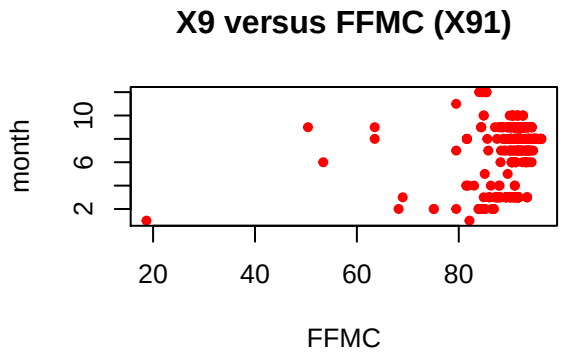


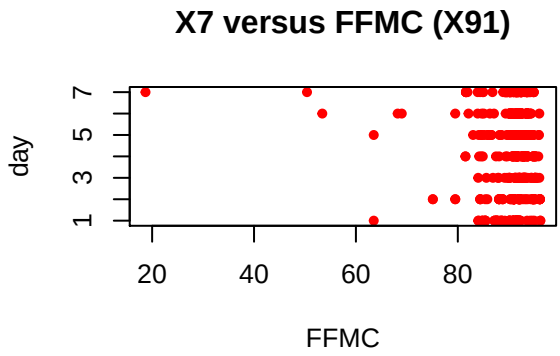
Vemos que ambas variables están relacionadas exponencialmente según la tendencia que presentan en sus datos. Si representamos X91 (FFMC) frente a las otras variables(excepto la salida):

```
par(mfrow=c(2,2))
plot(forestFires$X91,forestFires$X1,pch=20,col="red",xlab="FFMC", ylab = "X Coord in Park ",
     main = "X1 versus FFMC (X91)")
plot(forestFires$X91,forestFires$X3,pch=20,col="red",xlab="FFMC", ylab = "Y Coord in Park ",
     main = "X3 versus FFMC (X91)")
par(mfrow=c(2,2))
```

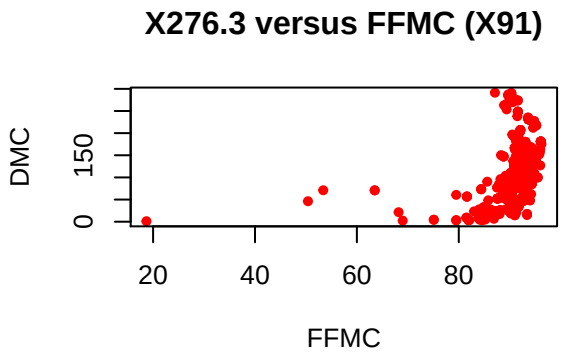


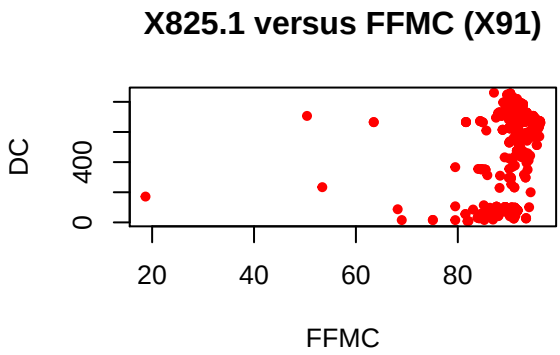

```
plot(forestFires$X91,forestFires$X9,pch=20,col="red",xlab="FFMC", ylab = "month",
     main = "X9 versus FFMC (X91)")
plot(forestFires$X91,forestFires$X7,pch=20,col="red",xlab="FFMC", ylab = "day",
     main = "X7 versus FFMC (X91)")
par(mfrow=c(2,2))
```



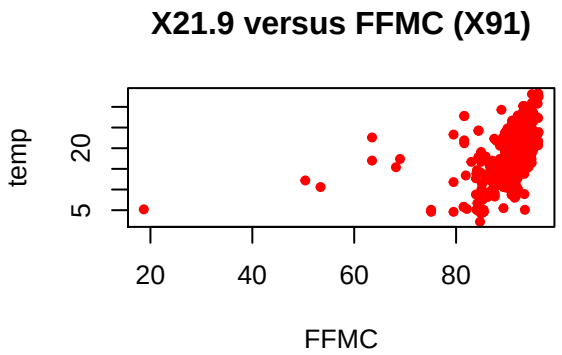


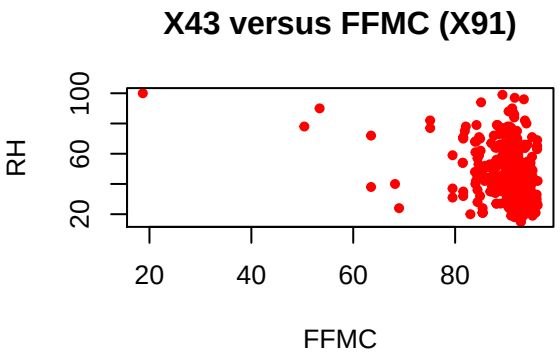

```
plot(forestFires$X91,forestFires$X276.3,pch=20,col="red",xlab="FFMC", ylab = "DMC",
     main = "X276.3 versus FFMC (X91)")
plot(forestFires$X91,forestFires$X825.1,pch=20,col="red",xlab="FFMC", ylab = "DC",
     main = "X825.1 versus FFMC (X91)")
par(mfrow=c(2,2))
```



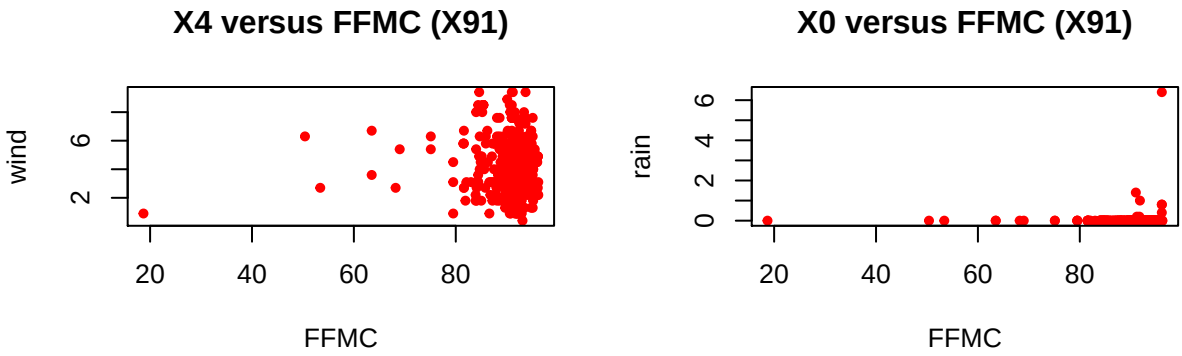



```
plot(forestFires$X91,forestFires$X21.9,pch=20,col="red",xlab="FFMC", ylab = "temp",
     main = "X21.9 versus FFMC (X91)")
plot(forestFires$X91,forestFires$X43,pch=20,col="red",xlab="FFMC", ylab = "RH",
     main = "X43 versus FFMC (X91)")
par(mfrow=c(2,2))
```





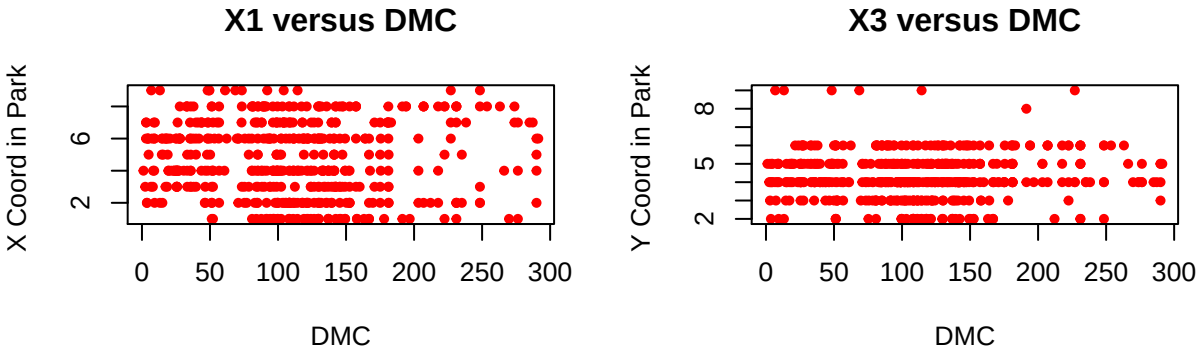

```
plot(forestFires$X91,forestFires$X4,pch=20,col="red",xlab="FFMC", ylab = "wind",
     main = "X4 versus FFMC (X91)")
plot(forestFires$X91,forestFires$X0,pch=20,col="red",xlab="FFMC", ylab = "rain",
     main = "X0 versus FFMC (X91)")
```



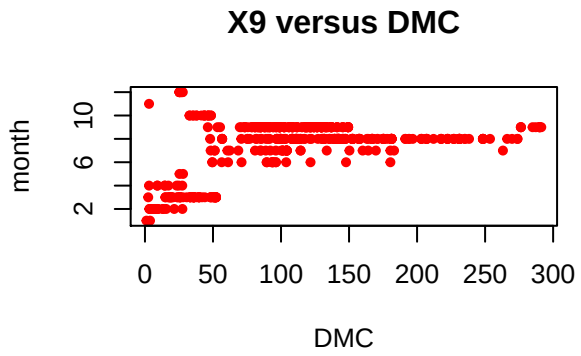
Podemos observar en las gráficas que X91 (el FFMC) parece estar relacionado también con las variables X276.3 (DMC) de forma exponencial, y también con X825.1 (DC) y X21.9 (temperatura), por tanto, si hubiera que hacer una selección de variables, quizás esta variable no la consideraríamos para el modelo. Tampoco consideraríamos X1 y X3, que son las variables que indican la coordenada X e Y respectivamente, dentro del parque Montesino donde se registran los incendios, pues no parecen guardar ninguna relación con la variable salida ni con ninguna otra. Seguramente también excluiríamos a la variable mes (X9), pues parece estar explicada por la variable temperatura, como hemos comentado antes.

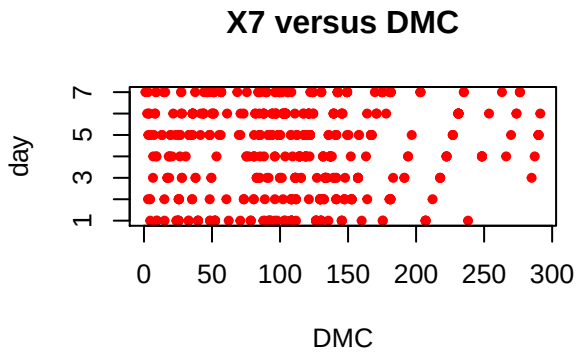
Dada la similitud de la variable DMC (X276.3) con FFMC (X91) descartada anteriormente, analizamos DMC más a fondo:

```
par(mfrow=c(2,2))
plot(forestFires$X276.3,forestFires$X1,pch=20,col="red",xlab="DMC", ylab = "X Coord in Park ",
     main = "X1 versus DMC")
plot(forestFires$X276.3,forestFires$X3,pch=20,col="red",xlab="DMC", ylab = "Y Coord in Park ",
     main = "X3 versus DMC")
par(mfrow=c(2,2))
```

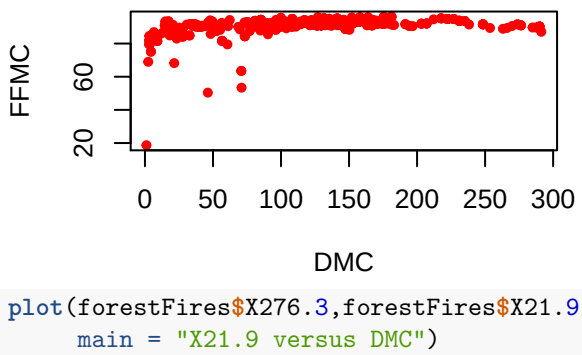


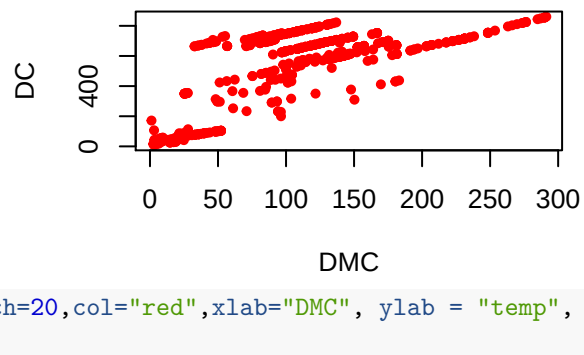

```
plot(forestFires$X276.3,forestFires$X9,pch=20,col="red",xlab="DMC", ylab = "month",
     main = "X9 versus DMC")
plot(forestFires$X276.3,forestFires$X7,pch=20,col="red",xlab="DMC", ylab = "day",
     main = "X7 versus DMC")
par(mfrow=c(2,2))
```



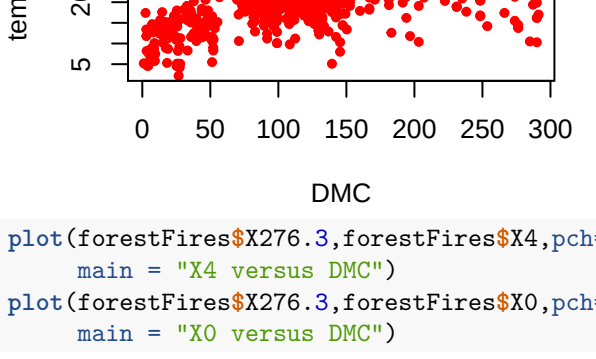


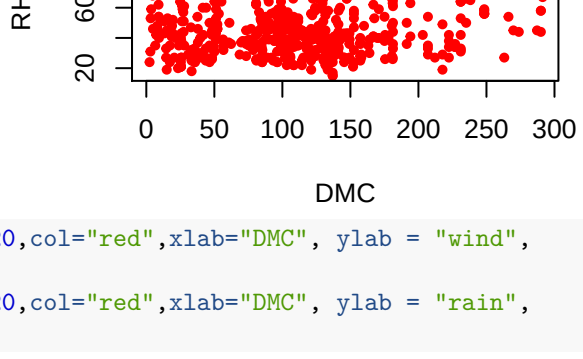

```
plot(forestFires$X276.3,forestFires$X91,pch=20,col="red",xlab="DMC", ylab = "FFMC",
     main = "X276.3 versus DMC")
plot(forestFires$X276.3,forestFires$X825.1,pch=20,col="red",xlab="DMC", ylab = "DC",
     main = "X825.1 versus DMC")
par(mfrow=c(2,2))
```



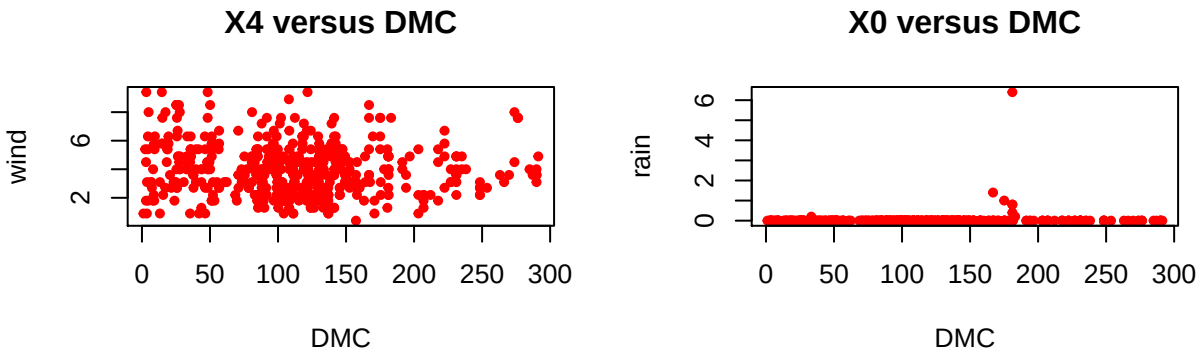



```
plot(forestFires$X276.3,forestFires$X21.9,pch=20,col="red",xlab="DMC", ylab = "temp",
     main = "X21.9 versus DMC")
plot(forestFires$X276.3,forestFires$X43,pch=20,col="red",xlab="DMC", ylab = "RH",
     main = "X43 versus DMC")
par(mfrow=c(2,2))
```






```
plot(forestFires$X276.3,forestFires$X4,pch=20,col="red",xlab="DMC", ylab = "wind",
     main = "X4 versus DMC")
plot(forestFires$X276.3,forestFires$X0,pch=20,col="red",xlab="DMC", ylab = "rain",
     main = "X0 versus DMC")
```



DMC (X276.3) representa el código de humedad (druff moisture code), medida utilizada para medir la humedad de las capas superficiales y profundas del suelo. Podemos observar que parece que tiene una relación con X9 (el mes), presentando valores de humedad del subsuelo más bajos en meses de invierno y otoño y más altos en verano. Esto tiene sentido pues, en los datos registrados, la mayoría de las lluvias se registraban en meses de verano, especialmente en agosto, quizá por tormentas de verano. También observamos que X276.3 o DMC presenta una relación logaritmica con la variable FFMC (X91), y también parece estar relacionada con DC (X825.1), variable que mide la sequía del suelo, pues parece que a mayor sequía en la capa superficial más humedad hay en las capas profundas. También parece que X276.3 guarda una relación polinómica con X21.9 (la temperatura). Podemos concluir entonces que, si tuviéramos que seleccionar variables para ajustar un modelo, quizás tampoco tendríamos en cuenta a DMC (X276.3) ni a DC (X825.1).

Conclusiones del EDA:

Aunque no hemos analizado las 13 variables intensamente, este EDA realizado nos ha servido para descubrir ciertas cosas sobre el dataset como que:

- la variable respuesta Área (X70.76) presenta un sesgo positivo muy grande y necesita una transformación

- Se producen más incendios en meses de verano (meses con temperaturas más altas)
- la mayoría de los incendios se producen en viernes y fines de semana
- X1 y X3 parecen ser independientes de todas las demás variables
- en la mayoría de los incendios no llovió
- Los meses de verano son los meses con más lluvias registradas, especialmente agosto
- Las variables mes (X9) y temperatura (temp) están relacionadas
- La variable FFMC (X91) está relacionada exponencialmente con la variable ISI (X7.1)
- La variable FFMC (X91) está relacionada con X276.3 (DMC) de forma exponencial, y también con X825.1 (DC) y X21.9 (temperatura)
- La variable DMC (X276.3) tiene una relación con X9 (el mes)
- La variable DMC (X276.3) presenta una relación logaritmica con la variable FFMC (X91)
- La variable DMC (X276.3) parece estar relacionada con DC (X825.1) y con X21.9 (la temperatura)
- La variable día (X7) no parece estar relacionada con ninguna otra variable

Aunque ninguna variable presenta a simple vista una relación directa con la variable respuesta Área (X70.76), podemos concluir que estamos ante un dataset que incluye bastantes variables relacionadas entre sí que se explican unas a otras y que algunas de las variables presentan sesgos y desplazamientos en su distribución, por lo que será importante transformar y normalizar algunas de estas variables. Además, contiene varias variables nominales como el día, mes o RH que será conveniente convertir en variables “dummy” o variables indicadoras, ya que pasar el array numérico tal cual indicaría que hay un orden entre las categorías de la variable que en este caso realmente no existe.

Apartado Regresión

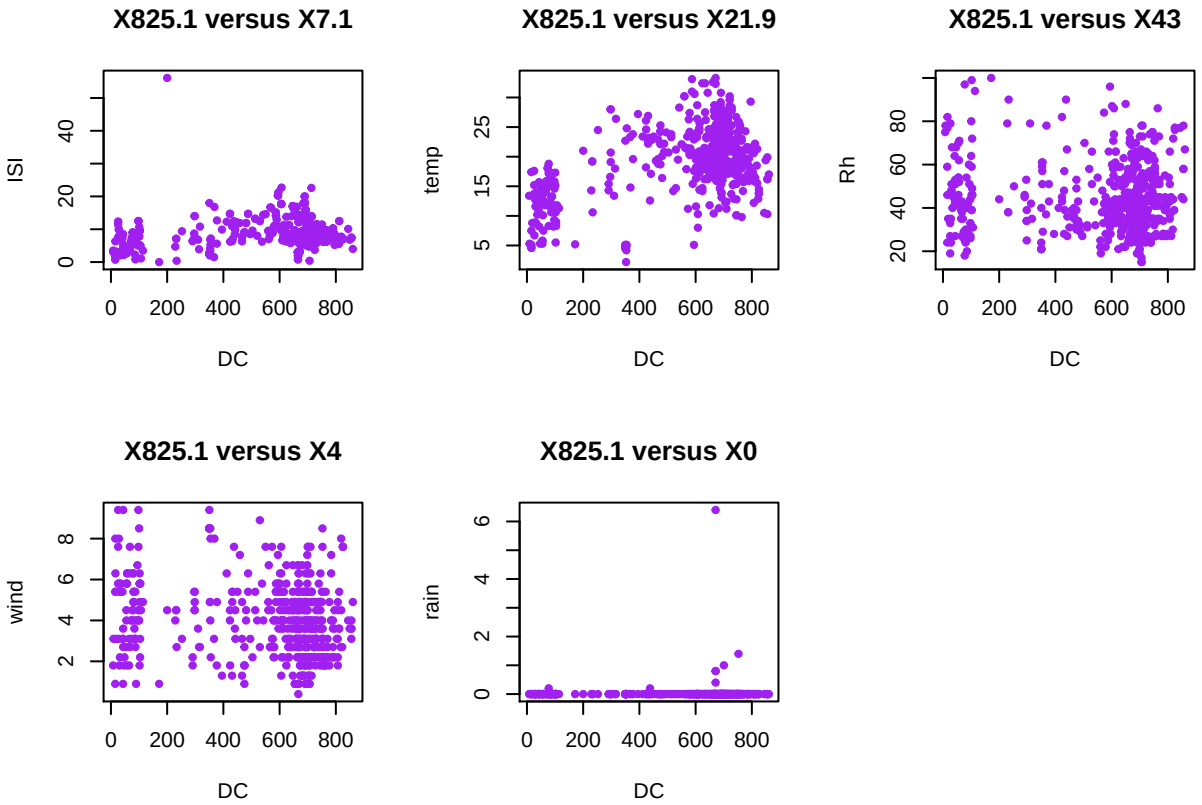
Regresión Lineal Simple

Dado el límite de 5 modelos lineales simples, se seleccionarán 5 de las 12 variables del dataset para ajustar cada uno de dichos modelos.

Remiténdonos al EDA realizado anteriormente, se descartarán las variables: - MES (X9) por su relación con temperatura y otras variables - FFMC (X91) por su relación clara con ISI (X7.1), X276.3 (DMC), X825.1 (DC) y temperatura (X21.9) - DMC (X276.3) por su relación con el MES (X9), FFMC (X91), DC (X825.1) y temperatura (X21.9) - X1 y X3 (cordenadas en el parque), pues no parecen aportar nada a la variable salida y a otras variables - DÍA (X7) pues tampoco parece aportar mucha información

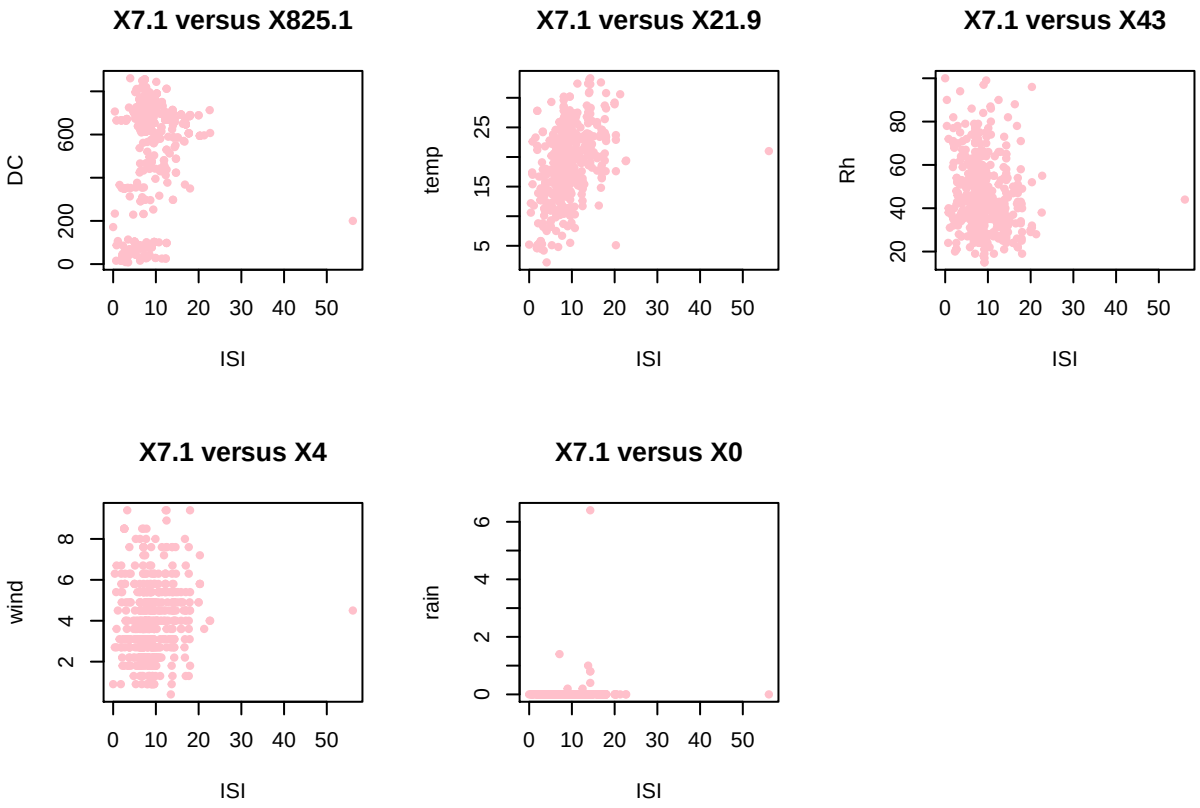
Partiendo de 6 variables descartadas, aún nos queda elegir 5 entre otras 6: DC (X825.1), ISI (X7.1), TEMP (X21.9),RH (X43), WIND (X4) y RAIN (X0). Comenzamos analizando DC (X825.1):

```
par(mfrow=c(2,3))
plot(forestFires$X825.1,forestFires$X7.1,pch=20,col="purple",xlab="DC", ylab = "ISI", main = "X825.1 ve
plot(forestFires$X825.1,forestFires$X21.9,pch=20,col="purple",xlab="DC", ylab = "temp", main = "X825.1 
plot(forestFires$X825.1,forestFires$X43,pch=20,col="purple",xlab="DC", ylab = "Rh", main = "X825.1 versu
plot(forestFires$X825.1,forestFires$X4,pch=20,col="purple",xlab="DC", ylab = "wind", main = "X825.1 vers
plot(forestFires$X825.1,forestFires$X0,pch=20,col="purple",xlab="DC", ylab = "rain", main = "X825.1 vers
```



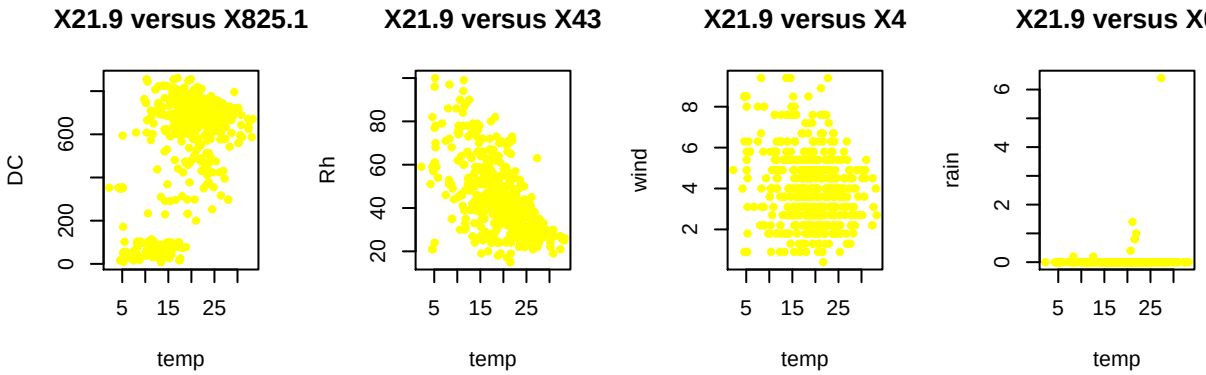
Vemos que parece tener una relación polinómica con ISI (X7.1). Analizamos ahora ISI (X7.1):

```
par(mfrow=c(2,3))
plot(forestFires$X7.1,forestFires$X825.1,pch=20,col="pink",xlab="ISI", ylab = "DC", main = "X7.1 versus 
plot(forestFires$X7.1,forestFires$X21.9,pch=20,col="pink",xlab="ISI", ylab = "temp", main = "X7.1 versus
plot(forestFires$X7.1,forestFires$X43,pch=20,col="pink",xlab="ISI", ylab = "Rh", main = "X7.1 versus X43
plot(forestFires$X7.1,forestFires$X4,pch=20,col="pink",xlab="ISI", ylab = "wind", main = "X7.1 versus X4
plot(forestFires$X7.1,forestFires$X0,pch=20,col="pink",xlab="ISI", ylab = "rain", main = "X7.1 versus X0
```



Observamos de nuevo la relación polinónica con DC y también una relación que podría modelarse como lineal con la temperatura (X21.9), por tanto, descartamos ISI (X7.1) para ajustar nuestros modelos lineales. Hemos seleccionado, por tanto, 5 variables de las cuales 4 representan condiciones meteorológicas: X21.9 (temperatura), X43 (Humedad relativa), X4 (viento) y X0 (lluvia), y DC que representa el código de sequía del suelo (X825.1). Si representamos la temperatura frente a las otras 4 variables seleccionadas:

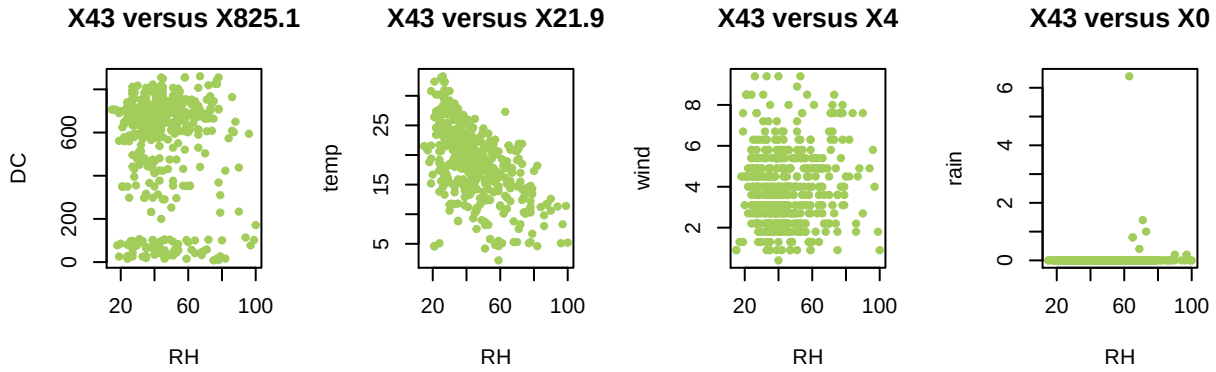
```
par(mfrow=c(2,4))
plot(forestFires$X21.9,forestFires$X825.1,pch=20,col="yellow",xlab="temp", ylab = "DC", main = "X21.9 v
plot(forestFires$X21.9,forestFires$X43,pch=20,col="yellow",xlab="temp", ylab = "Rh", main = "X21.9 versu
plot(forestFires$X21.9,forestFires$X4,pch=20,col="yellow",xlab="temp", ylab = "wind", main = "X21.9 vers
plot(forestFires$X21.9,forestFires$X0,pch=20,col="yellow",xlab="temp", ylab = "rain", main = "X21.9 vers
```



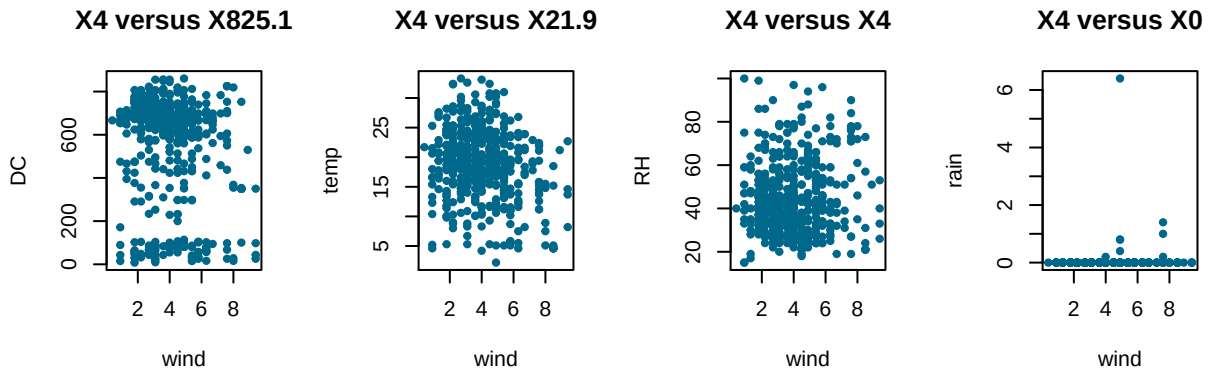
Observamos que no hay relaciones evidentes entre las variables, si bien sí que parece que a temperaturas más altas menor es la humedad relativa. Podemos comprobar que lo mismo ocurre con la humedad relativa, el viento y la lluvia:

```
par(mfrow=c(2,4))
plot(forestFires$X43,forestFires$X825.1,pch=20,col="darkolivegreen3",xlab="RH", ylab = "DC", main = "X43
```

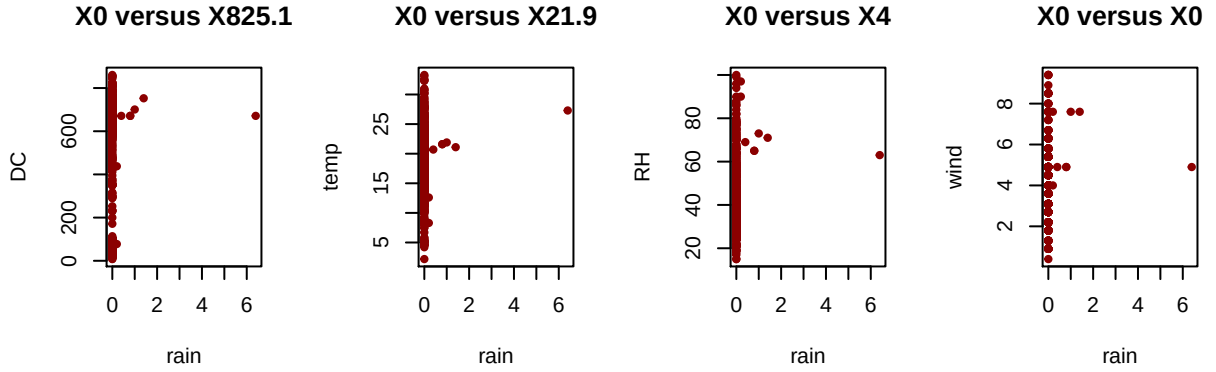
```
plot(forestFires$X43,forestFires$X21.9,pch=20,col="darkolivegreen3",xlab="RH", ylab = "temp", main = "X4
plot(forestFires$X43,forestFires$X4,pch=20,col="darkolivegreen3",xlab="RH", ylab = "wind", main = "X43 
plot(forestFires$X43,forestFires$X0,pch=20,col="darkolivegreen3",xlab="RH", ylab = "rain", main = "X43 
par(mfrow=c(2,4))
```




```
plot(forestFires$X4,forestFires$X825.1,pch=20,col="deepskyblue4",xlab="wind", ylab = "DC", main = "X4 ve
plot(forestFires$X4,forestFires$X21.9,pch=20,col="deepskyblue4",xlab="wind", ylab = "temp", main = "X4 
plot(forestFires$X4,forestFires$X43,pch=20,col="deepskyblue4",xlab="wind", ylab = "RH", main = "X4 versu
plot(forestFires$X4,forestFires$X0,pch=20,col="deepskyblue4",xlab="wind", ylab = "rain", main = "X4 vers
par(mfrow=c(2,4))
```




```
plot(forestFires$X0,forestFires$X825.1,pch=20,col="darkred",xlab="rain", ylab = "DC", main = "X0 versus 
plot(forestFires$X0,forestFires$X21.9,pch=20,col="darkred",xlab="rain", ylab = "temp", main = "X0 versus
plot(forestFires$X0,forestFires$X43,pch=20,col="darkred",xlab="rain", ylab = "RH", main = "X0 versus X4
plot(forestFires$X0,forestFires$X4,pch=20,col="darkred",xlab="rain", ylab = "wind", main = "X0 versus X
```



Ya que tenemos las 5 variables seleccionadas sobre las que hacer nuestro modelo de regresión, aplicamos un pequeño preprocesamiento en el que convertiremos las variables nominales en variables "dummy" o indicativas:

```
mes.f = factor(forestFires$X9)
dummiesmes = model.matrix(~mes.f)
dummiesmes=dummiesmes[,-1]
forestFires$X9<-dummiesmes

dia.f = factor(forestFires$X7)
dummiesdia = model.matrix(~dia.f)
dummiesdia=dummiesdia[,-1]
forestFires$X7<-dummiesdia
```

Dividimos el dataset en un conjunto de train y otro de test. Para train dejaremos el 80% del dataset mientras que reservaremos el 20% para test. Con el fin de que los resultados sean replicables, establecemos una semilla antes de partir los datos. Una vez partidos los datos, ajustamos los 5 modelos:

```
set.seed(1)
trainIndex<-as.integer(nrow(forestFires)*0.80,replace=FALSE)
randomdata<-sample(1:nrow(forestFires),trainIndex)
train <- forestFires[randomdata,]
test <- forestFires[-randomdata,]

fit1=lm(log1p(X70.76)~X825.1,data=train)
fit2=lm(log1p(X70.76)~X21.9,data=train)
fit3=lm(log1p(X70.76)~X43,data=train)
fit4=lm(log1p(X70.76)~X4,data=train)
fit5=lm(log1p(X70.76)~X0,data=train)
summary(fit1)
```

```
##
## Call:
## lm(formula = log1p(X70.76) ~ X825.1, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.159 -1.115 -0.711  0.907  5.876
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.9740027  0.1692657   5.754 1.71e-08 ***
## X825.1      0.0002162  0.0002802   0.772    0.441
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.389 on 410 degrees of freedom
## Multiple R-squared:  0.00145,    Adjusted R-squared:  -0.0009851
## F-statistic: 0.5955 on 1 and 410 DF,  p-value: 0.4407
```

```
summary(fit2)
```

```
##
## Call:
## lm(formula = log1p(X70.76) ~ X21.9, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3053 -1.0854 -0.7217  0.9274  5.8067
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.78891    0.23480   3.360 0.000853 ***
## X21.9        0.01594    0.01176   1.356 0.175917
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.387 on 410 degrees of freedom
## Multiple R-squared:  0.004463,   Adjusted R-squared:  0.002035
## F-statistic: 1.838 on 1 and 410 DF,  p-value: 0.1759
```

```
summary(fit3)
```

```
##
## Call:
## lm(formula = log1p(X70.76) ~ X43, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2168 -1.0997 -0.7065  0.9414  5.8290
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.279596   0.197244   6.487 2.52e-10 ***
## X43         -0.004183   0.004158  -1.006    0.315
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.388 on 410 degrees of freedom
## Multiple R-squared:  0.002462,   Adjusted R-squared:  2.922e-05
## F-statistic: 1.012 on 1 and 410 DF,  p-value: 0.315
```

```
summary(fit4)
```

```
##
## Call:
## lm(formula = log1p(X70.76) ~ X4, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2384 -1.0800 -0.7072  0.9445  5.9012
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.95047    0.16730   5.681 2.54e-08 ***
## X4           0.03599    0.03842   0.937     0.35
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.388 on 410 degrees of freedom
## Multiple R-squared:  0.002135,   Adjusted R-squared:  -0.0002987
## F-statistic: 0.8773 on 1 and 410 DF,  p-value: 0.3495
```

```
summary(fit5)
```

```
##
## Call:
## lm(formula = log1p(X70.76) ~ X0, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1030 -1.1030 -0.6055  0.9215  5.8926
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.10305    0.06877  16.040   <2e-16 ***
## X0          -0.82129    0.65948  -1.245    0.214
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.387 on 410 degrees of freedom
## Multiple R-squared:  0.003769,   Adjusted R-squared:  0.001339
## F-statistic: 1.551 on 1 and 410 DF,  p-value: 0.2137
```

Podemos observar que para los 5 modelos obtenemos un RSE cercano a 0, lo cual indica que el modelo se ajusta bastante bien a los datos (quizás demasiado bien). También para todos los modelos se obtiene un R^2 muy bajo cercano a 0, lo cual denota que lejos de seguir una tendencia lineal, los puntos más bien son aleatorios, además en todos los casos obtenemos un p-valor superior a 0.05 en los 5 modelos, por lo que no podemos rechazar la hipótesis nula de que el coeficiente real sea 0.

Por último, calculamos el error en test para cada uno de los 5 modelos creados. Como a la variable respuesta (X70.76) le aplicamos una transformación logaritmica (log(x+1)), a la salida del modelo le aplicamos la inversa de dicha función ($e^x - 1$):

```
yprime1=predict(fit1,test)
yprime1=expm1(yprime1)
yprime2=predict(fit2,test)
yprime2=expm1(yprime2)
yprime3=predict(fit3,test)
yprime3=expm1(yprime3)
yprime4=predict(fit4,test)
yprime4=expm1(yprime4)
yprime5=predict(fit5,test)
yprime5=expm1(yprime5)
#cálculo RMSE
mse1=sqrt(sum(abs(test$X70.76-yprime1)^2)/length(yprime1))
mse2=sqrt(sum(abs(test$X70.76-yprime2)^2)/length(yprime2))
mse3=sqrt(sum(abs(test$X70.76-yprime3)^2)/length(yprime3))
mse4=sqrt(sum(abs(test$X70.76-yprime4)^2)/length(yprime4))
mse5=sqrt(sum(abs(test$X70.76-yprime5)^2)/length(yprime5))

mse1
```

```
## [1] 30.89161
```

```
mse2
```

```
## [1] 30.91041
```

```
mse3
```

```
## [1] 30.88551
```

```
mse4
```

```
## [1] 30.91213
```

```
mse5
```

```
## [1] 30.91003
```

Observamos que sobre el mismo conjunto de train y test, los 5 modelos obtienen RMSE muy parecido, si bien difieren en algunas décimas. En general, los 5 modelos obtenidos son bastante malos, deduciendo que no se puede modelar el comportamiento de la variable salida con sólo una variable. Aunque ningún modelo es bueno, de los 5 modelos podemos decir que quizás el menos malo es el de la variable X21.9 (la temperatura), ya que tiene el mayor Rš y R^2 Ajustado, y el menor p-valor, a pesar de que su RMSE es unas décimas más elevado para test que otros modelos.

Regresión Lineal Múltiple

Usaremos los datos de test y train particionados para el apartado anterior. Comenzamos ajustando una regresión múltiple sobre todas las variables:

```
fitall<-lm(log1p(X70.76)~.,train)
summary(fitall)
```

```
##
## Call:
## lm(formula = log1p(X70.76) ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0793 -0.9713 -0.4618  0.8504  5.1239
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.383930   1.747390  -0.792   0.4288
## X1           0.041101   0.035690   1.152   0.2502
## X3           0.025124   0.067883   0.370   0.7115
## X9mes.f2    -0.025425   1.923083  -0.013   0.9895
## X9mes.f3    -0.843568   1.956473  -0.431   0.6666
## X9mes.f4    -0.533722   1.973190  -0.270   0.7869
## X9mes.f5     0.004227   2.173057   0.002   0.9984
## X9mes.f6    -0.689786   1.920303  -0.359   0.7196
## X9mes.f7    -0.497264   1.976305  -0.252   0.8015
## X9mes.f8     0.020885   2.017299   0.010   0.9917
## X9mes.f9     0.527069   2.073057   0.254   0.7994
## X9mes.f10    0.268990   2.145505   0.125   0.9003
## X9mes.f11          NA         NA      NA       NA
## X9mes.f12    2.435645   2.081286   1.170   0.2426
## X7dia.f2     0.063627   0.279947   0.227   0.8203
## X7dia.f3    -0.082968   0.287666  -0.288   0.7732
## X7dia.f4     0.032685   0.276463   0.118   0.9060
## X7dia.f5    -0.077896   0.253075  -0.308   0.7584
## X7dia.f6     0.277975   0.251158   1.107   0.2691
## X7dia.f7    -0.019015   0.250406  -0.076   0.9395
## X91          0.015368   0.020559   0.748   0.4552
## X276.3       0.002814   0.002105   1.337   0.1820
```

```
## X825.1      -0.002192   0.001573  -1.394   0.1642
## X7.1        -0.011514   0.020260  -0.568   0.5702
## X21.9        0.057847   0.025210   2.295   0.0223 *
## X43          0.008496   0.007365   1.154   0.2494
## X4           0.052981   0.043648   1.214   0.2256
## X0          -1.131188   0.703691  -1.608   0.1088
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.376 on 385 degrees of freedom
## Multiple R-squared:  0.07928,    Adjusted R-squared:  0.0171
## F-statistic: 1.275 on 26 and 385 DF,  p-value: 0.1686
```

Vemos que obtenemos un modelo muy malo de RSE= 1.376, pero con un R^2 ajustado de 0.0171 y un p-valor muy superior a 0.05. Calculamos su error de test. Como parece que la temperatura (X21.9) es una variable influyente en la salida y que su coeficiente es distinto de 0, comenzamos a construir el modelo partiendo de esta variable. Como desde uun principio sabemos que los datos no presentan interacciones lineales, hacemos uso de la no linealidad:

```
fittmp<-lm(log1p(X70.76)~X21.9+I(X21.9^2),train)
summary(fitall)
```

```
##
## Call:
## lm(formula = log1p(X70.76) ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0793 -0.9713 -0.4618  0.8504  5.1239
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.383930   1.747390  -0.792   0.4288
## X1           0.041101   0.035690   1.152   0.2502
## X3           0.025124   0.067883   0.370   0.7115
## X9mes.f2    -0.025425   1.923083  -0.013   0.9895
## X9mes.f3    -0.843568   1.956473  -0.431   0.6666
## X9mes.f4    -0.533722   1.973190  -0.270   0.7869
## X9mes.f5     0.004227   2.173057   0.002   0.9984
## X9mes.f6    -0.689786   1.920303  -0.359   0.7196
## X9mes.f7    -0.497264   1.976305  -0.252   0.8015
## X9mes.f8     0.020885   2.017299   0.010   0.9917
## X9mes.f9     0.527069   2.073057   0.254   0.7994
## X9mes.f10    0.268990   2.145505   0.125   0.9003
## X9mes.f11          NA         NA      NA       NA
## X9mes.f12    2.435645   2.081286   1.170   0.2426
## X7dia.f2     0.063627   0.279947   0.227   0.8203
## X7dia.f3    -0.082968   0.287666  -0.288   0.7732
## X7dia.f4     0.032685   0.276463   0.118   0.9060
## X7dia.f5    -0.077896   0.253075  -0.308   0.7584
## X7dia.f6     0.277975   0.251158   1.107   0.2691
## X7dia.f7    -0.019015   0.250406  -0.076   0.9395
## X91          0.015368   0.020559   0.748   0.4552
## X276.3       0.002814   0.002105   1.337   0.1820
## X825.1      -0.002192   0.001573  -1.394   0.1642
```

```
## X7.1        -0.011514   0.020260  -0.568   0.5702
## X21.9        0.057847   0.025210   2.295   0.0223 *
## X43          0.008496   0.007365   1.154   0.2494
## X4           0.052981   0.043648   1.214   0.2256
## X0          -1.131188   0.703691  -1.608   0.1088
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.376 on 385 degrees of freedom
## Multiple R-squared:  0.07928,    Adjusted R-squared:  0.0171
## F-statistic: 1.275 on 26 and 385 DF,  p-value: 0.1686
```

Dado el bajo R^2 ajustado y el alto p-valor, el modelo no nos sirve. Continuamos añadiendo otras variables:

```
fittmp<-lm(log1p(X70.76)~X21.9+I(X21.9^2)+X825.1+X43+X4+X0,train)
summary(fitall)
```

```
##
## Call:
## lm(formula = log1p(X70.76) ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0793 -0.9713 -0.4618  0.8504  5.1239
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.383930   1.747390  -0.792   0.4288
## X1           0.041101   0.035690   1.152   0.2502
## X3           0.025124   0.067883   0.370   0.7115
## X9mes.f2    -0.025425   1.923083  -0.013   0.9895
## X9mes.f3    -0.843568   1.956473  -0.431   0.6666
## X9mes.f4    -0.533722   1.973190  -0.270   0.7869
## X9mes.f5     0.004227   2.173057   0.002   0.9984
## X9mes.f6    -0.689786   1.920303  -0.359   0.7196
## X9mes.f7    -0.497264   1.976305  -0.252   0.8015
## X9mes.f8     0.020885   2.017299   0.010   0.9917
## X9mes.f9     0.527069   2.073057   0.254   0.7994
## X9mes.f10    0.268990   2.145505   0.125   0.9003
## X9mes.f11          NA         NA      NA       NA
## X9mes.f12    2.435645   2.081286   1.170   0.2426
## X7dia.f2     0.063627   0.279947   0.227   0.8203
## X7dia.f3    -0.082968   0.287666  -0.288   0.7732
## X7dia.f4     0.032685   0.276463   0.118   0.9060
## X7dia.f5    -0.077896   0.253075  -0.308   0.7584
## X7dia.f6     0.277975   0.251158   1.107   0.2691
## X7dia.f7    -0.019015   0.250406  -0.076   0.9395
## X91          0.015368   0.020559   0.748   0.4552
## X276.3       0.002814   0.002105   1.337   0.1820
## X825.1      -0.002192   0.001573  -1.394   0.1642
## X7.1        -0.011514   0.020260  -0.568   0.5702
## X21.9        0.057847   0.025210   2.295   0.0223 *
## X43          0.008496   0.007365   1.154   0.2494
## X4           0.052981   0.043648   1.214   0.2256
## X0          -1.131188   0.703691  -1.608   0.1088
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.376 on 385 degrees of freedom
## Multiple R-squared:  0.07928,    Adjusted R-squared:  0.0171
## F-statistic: 1.275 on 26 and 385 DF,  p-value: 0.1686
```

Obtenemos similar R^2 y p-valor al modelo anterior. Vamos pensando y probando interaciones, hasta que encontramos un modelo que usa interacciones y no linealinealidad con un p-valor bajo:

```
fit1Rlm<-lm(log1p(X70.76)~X21.9+I(X21.9^2)+I(X21.9^2*X43)+I(X0*X43)+I(X4)+X43*X825.1,train)
#fit1Rlm<-lm(log1p(X70.76)~X21.9+I(X21.9^2)+I(X21.9^2*X43)+I(X0^2*X43^2)+I(X4^2)+X43*X825.1,train)
#fit1Rlm<-lm(log1p(X70.76)~X21.9+I(X21.9^2)+I(X21.9^2*X43)+I(X0^2*X43^2)+I(X4^2)+X43*X825.1,train)

#fit1Rlm<-lm(log1p(X70.76)~X21.9+I(X21.9^2)+I(X21.9^2*X43)+I(X0^2*X43^2)+I(X4^2)+X43*X825.1,train)

summary(fit1Rlm)
```

```
##
## Call:
## lm(formula = log1p(X70.76) ~ X21.9 + I(X21.9^2) + I(X21.9^2 *
##     X43) + I(X0 * X43) + I(X4) + X43 * X825.1, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8272 -1.0046 -0.5562  0.7696  5.8302
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        2.686e+00  1.038e+00   2.587  0.01004 *
## X21.9             -2.344e-01  7.384e-02  -3.175  0.00162 **
## I(X21.9^2)         4.011e-03  1.545e-03   2.597  0.00976 **
## I(X21.9^2 * X43)   7.259e-05  2.910e-05   2.494  0.01302 *
## I(X0 * X43)       -1.833e-02  9.936e-03  -1.845  0.06574 .
## I(X4)              5.511e-02  4.173e-02   1.321  0.18736
## X43               -6.802e-03  1.109e-02  -0.613  0.53989
## X825.1             1.468e-03  9.035e-04   1.625  0.10494
## X43:X825.1        -2.338e-05  1.780e-05  -1.313  0.18991
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.367 on 403 degrees of freedom
## Multiple R-squared:  0.04962,    Adjusted R-squared:  0.03075
## F-statistic:  2.63 on 8 and 403 DF,  p-value: 0.008108
```

Usando las 5 variables seleccionadas, encontamos este modelo con un R^2 ajustado de 0.03075, superior al del anterior y lo más alto encontrado por ahora, y además tenemos un p-valor de 0.0081 que es inferior a 0.05, por lo que se rechaza la hipótesis nula. Si calculamos su error para test:

```
yprime=predict(fit1Rlm,test)
yprime=expm1(yprime)
sqrt(sum(abs(test$X70.76-yprime)^2)/length(yprime))
```

```
## [1] 30.94152
```

Obtenemos un error muy similar al que obtuvimos con los modelos lineales simples del primer apartado. Al no tener una relación clara, estos datos son difícles de modelar.

k-NN para regresión no paramétrica

Comenzamos con el mejor modelo encontrado en el apartado anterior. Leemos las particiones tal y como hemos hecho en clase, le aplicamos un poco de preprocesamiento a los datos, como hemos hecho en apartados anteriores y utilizamos el modelo del apartado anterior para ajustar:

```
setwd("/home/cris/mrcrstnherediagmez@gmail.com/Intro_Ciencia_de_datos/Trabajo Final/Datasets Regresion/f

nombre <- "forestFires"
run_lm_fold <- function(i, x, tt = "test") {
file <- paste(x, "-5-", i, "tra.dat", sep="")
x_tra <- read.csv(file, comment.char="@")
file <- paste(x, "-5-", i, "tst.dat", sep="")
x_tst <- read.csv(file, comment.char="@")
In <- length(names(x_tra)) - 1
names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
names(x_tra)[In+1] <- "Y"
names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
names(x_tst)[In+1] <- "Y"
# creando variables indicativas para mes y día
mes.f.tra = factor(x_tra$X3)
mes.f.tst = factor(x_tst$X3)
dummiesmes.tra = model.matrix(~mes.f.tra)
dummiesmes.tst = model.matrix(~mes.f.tst)
dummiesmes.tra=dummiesmes.tra[,-1]
dummiesmes.tst=dummiesmes.tst[,-1]
x_tra$X3<-dummiesmes.tra
x_tst$X3<-dummiesmes.tst

dia.f.tra = factor(x_tra$X4)
dia.f.tst = factor(x_tst$X4)
dummiesdia.tra = model.matrix(~dia.f.tra)
dummiesdia.tst = model.matrix(~dia.f.tst)
dummiesdia.tra=dummiesdia.tra[,-1]
dummiesdia.tst=dummiesdia.tst[,-1]
x_tra$X4<-dummiesdia.tra
x_tst$X4<-dummiesdia.tst

if (tt == "train") {
  test <- x_tra
}else{
  test <- x_tst
}
fitMulti=lm(log1p(Y)~X9+I(X9^2)+I(X9^2*X10)+I(X12*X10)+I(X11)+X10*X7,x_tra)
yprime=predict(fitMulti,test)
yprime=expm1(yprime)
sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))
```

Vemos el MSE que obtenemos para train y test con el modelo de Reresión múltiple:

```
lmMSEtrain
```

```
## [1] 4156.586
```

```
lmMSEtest
```

```
## [1] 4167.963
```

Ahora repetimos el mismo procedimiento, pero usando knn en lugar de regresión múltiple para las mismas particiones:

```
setwd("/home/cris/mrcrstnherediagmez@gmail.com/Intro_Ciencia_de_datos/Trabajo Final/Datasets Regresion/

require("kknn")
```

```
## Loading required package: kknn
```

```
nombre <- "forestFires"
run_lm_fold <- function(i, x, tt = "test") {
file <- paste(x, "-5-", i, "tra.dat", sep="")
x_tra <- read.csv(file, comment.char="@")
file <- paste(x, "-5-", i, "tst.dat", sep="")
x_tst <- read.csv(file, comment.char="@")
In <- length(names(x_tra)) - 1
names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
names(x_tra)[In+1] <- "Y"
names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
names(x_tst)[In+1] <- "Y"

if (tt == "train") {
  test <- x_tra
}else{
  test <- x_tst
}
fitMulti=kknn(log1p(Y)~X9+I(X9^2)+I(X9^2*X10)+I(X12*X10)+I(X11)+X10*X7,x_tra,test)
yprime=fitMulti$fitted.values
yprime=expm1(yprime)
sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))
```

```
lmMSEtrain
```

```
## [1] 3821.7
```

```
lmMSEtest
```

```
## [1] 4150.948
```

y comparamos. Vemos que para train obtuvimos un MSE en train de 4156.586 para regresión múltiple, frente a 3821.7 obtenido con knn. Para MSE en test, obtenemos que los dos algoritmos rinden de forma muy similar, obteniendo con regresión múltiple MSE en test=4167.963 y 4150.948 con knn. Por tanto, ante los datos podemos decir que parece que knn redce el error en test levemente con respecto al otro algoritmo.

4. Comparativa de resultados

En el apartado anterior no se usaron todas las variables para ajustar los modelos de regresión múltiple y knn, por tanto, repetimos el proceso usando todas las variables. Además como las comparaciones han de ser genéricas e incluyen al algoritmo M5, que no sabemos si aplica transformación o no en la variable response,

tampoco aplicaremos dicha transformación a dicha variable en RM y knn. Finalmente anotamos las salidas en *regr_test_alumnos.csv* y *regr_train_alumnos.csv*.

```
setwd("/home/cris/mrcrstnherediagmez@gmail.com/Intro_Ciencia_de_datos/Trabajo Final/Datasets Regresion/
```

```
nombre <- "forestFires"
run_lm_fold <- function(i, x, tt = "test") {
file <- paste(x, "-5-", i, "tra.dat", sep="")
x_tra <- read.csv(file, comment.char="@")
file <- paste(x, "-5-", i, "tst.dat", sep="")
x_tst <- read.csv(file, comment.char="@")
In <- length(names(x_tra)) - 1
names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
names(x_tra)[In+1] <- "Y"
names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
names(x_tst)[In+1] <- "Y"

if (tt == "train") {
  test <- x_tra
}else{
  test <- x_tst
}
fitMulti=lm(Y~.,x_tra)
yprime=predict(fitMulti,test)
#yprime=expm1(yprime)
sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))
lmMSEtrain
```

```
## [1] 3945
```

```
lmMSEtest
```

```
## [1] 4060.943
```

```
setwd("/home/cris/mrcrstnherediagmez@gmail.com/Intro_Ciencia_de_datos/Trabajo Final/Datasets Regresion/
```

```
require("kknn")
nombre <- "forestFires"
run_lm_fold <- function(i, x, tt = "test") {
file <- paste(x, "-5-", i, "tra.dat", sep="")
x_tra <- read.csv(file, comment.char="@")
file <- paste(x, "-5-", i, "tst.dat", sep="")
x_tst <- read.csv(file, comment.char="@")
In <- length(names(x_tra)) - 1
names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
names(x_tra)[In+1] <- "Y"
names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
names(x_tst)[In+1] <- "Y"

if (tt == "train") {
  test <- x_tra
}else{
  test <- x_tst
```

```
}
fitMulti=kknn(Y~.,x_tra,test)
yprime=fitMulti$fitted.values
#yprime=expm1(yprime)
sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))
lmMSEtrain
```

```
## [1] 2206.328
```

```
lmMSEtest
```

```
## [1] 5840.684
```

Una vez que tenemos los datos anotados, leemos las tablas:

```
setwd("/home/cris/mrcrstnherediagmez@gmail.com/Intro_Ciencia_de_datos/Trabajo Final/")
#errores medios de test
resultados <- read.csv("regr_test_alumnos.csv")
tablatst <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatst) <- resultados[,1]

#errores medios de train
resultados <- read.csv("regr_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatra) <- resultados[,1]
```

y aplicamos el test de Wilcoxon para comparar los algoritmos de knn y MR:

```
difs <- (tablatst[,1] - tablatst[,2]) / tablatst[,1]
wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0, abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2) <- c(colnames(tablatst)[1], colnames(tablatst)[2])
head(wilc_1_2)
```

```
##      out_test_lm out_test_kknn
## [1,]   0.1909091     0.1000000
## [2,]   0.1000000     1.0294118
## [3,]   0.1000000     0.4339071
## [4,]   0.1000000     0.3885965
## [5,]   0.1548506     0.1000000
## [6,]   0.1000000     0.3061057
```

```
# Se aplica el test
LMvsKNNtst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative = "two.sided", paired=TRUE)
Rmas <- LMvsKNNtst$statistic
pvalue <- LMvsKNNtst$p.value
LMvsKNNtst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative = "two.sided", paired=TRUE)
Rmenos <- LMvsKNNtst$statistic
Rmas
```

```
##  V
## 78
```

```
Rmenos
```

```
##  V
## 93
```

```
pvalue
```

```
## [1] 0.7660294
```

Obtenemos un R^+ de 78, un R^- de 93 y un p-valor de 0.7660294, que a un nivel significativo de 0.05 podemos decir que no son distintos, o dicho de otra forma, hay un (1-0.7660294)*100 = 23.4% de confianza en que sean distintos. La diferencia no es significativa.

Comparativa múltiple de algoritmos

Comparamos los dos algoritmos anteriores con M5, usando Friedman y Holms. para ello usaremos los datos de tabla test:

```
test_friedman <- friedman.test(as.matrix(tablatst))
test_friedman$p.value
```

```
## [1] 0.01466602
```

Obtenenos un p-valor muy bajo de 0.01466602, por lo que opdemos afirmar a un nivel significativo de 0.05 que existen diferencias en, al menos un par de algoritmos (M5 con algún otro). Aplicamos por pares el test de post-hoc Holm:

```
tam <- dim(tablatra)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatra), groups, p.adjust = "holm", paired = TRUE)
```

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatra) and groups
##
##   1      2
## 2 0.0031 -
## 3 0.0032 0.0032
##
## P value adjustment method: holm
```

concluyendo que para 3 vs 1 y 3 vs 2 obtienen el mismo valor 0.0032, por lo que no hay diferencias significativas a favor de M5. Los otros dos pueden ser considerados equivalentes, ya que 1 vs 2 = 2 vs 1 = 0.0031 y 1 vs 3 = 2 vs 3 = 0.0032.

Si observamos los datos, de las tablas, podemos ver que los mejores resultados de train se obtienen para KNN con un error de 2206.328, sin embargo los mejores resultados en test los da RM con un error de 4060.943, mientras que los peores los da KNN con un error de 5840.684. Podemos deducir por tanto que KNN está sobreajustando los datos, de tal forma que el error de train obtenido es el más bajo pero en test se dispara. Sin considerar KNN por su problema de sobreajuste, según las comparativas el mejor modelo sería MR, pues obtiene, tanto en train como en test un error más bajo que M5, y tiene un buen balance de error train-test, por lo que no parece estar sobreajustando.

```
tablatra[11,]
```

```
##             out_train_lm out_train_kknn out_train_m5p
## forestFires         3945       2206.328          3980
```

```
tablatst[11,]
```

```
##             out_test_lm out_test_kknn out_test_m5p
## forestFires    4060.943      5840.684      4071.04
```

Análisis de los datos de Clasificación

Para clasificación se me asignó el conjunto de datos **tae**. ### Cálculo de estadísticos y descripción de los datos

```
setwd("/home/cris/mrcrstnherediagmez@gmail.com/Intro_Ciencia_de_datos/Trabajo Final/Datasets Clasificaci
tae<-read.csv("tae.dat", comment.char="@")
str(tae)
```

```
## 'data.frame':    150 obs. of  6 variables:
##  $ X1  : int  2 1 1 2 2 2 2 1 2 2 ...
##  $ X23 : int  15 23 5 7 23 9 10 22 15 10 ...
##  $ X3  : int  3 3 2 11 3 5 3 3 3 22 ...
##  $ X1.1: int  1 2 2 2 1 2 2 1 1 2 ...
##  $ X19 : int  17 49 33 55 20 19 27 58 20 9 ...
##  $ X3.1: int  3 3 3 3 3 3 3 3 3 3 ...
```

```
summary(tae)
```

```
##        X1             X23              X3             X1.1      
##  Min.   :1.000   Min.   : 1.00   Min.   : 1.00   Min.   :1.000  
##  1st Qu.:2.000   1st Qu.: 8.00   1st Qu.: 3.00   1st Qu.:2.000  
##  Median :2.000   Median :13.00   Median : 4.50   Median :2.000  
##  Mean   :1.813   Mean   :13.58   Mean   : 8.14   Mean   :1.853  
##  3rd Qu.:2.000   3rd Qu.:20.00   3rd Qu.:15.00   3rd Qu.:2.000  
##  Max.   :2.000   Max.   :25.00   Max.   :26.00   Max.   :2.000  
##       X19             X3.1      
##  Min.   : 3.00   Min.   :1.000  
##  1st Qu.:19.00   1st Qu.:1.000  
##  Median :27.00   Median :2.000  
##  Mean   :27.93   Mean   :2.013  
##  3rd Qu.:37.00   3rd Qu.:3.000  
##  Max.   :66.00   Max.   :3.000
```

Se trata de un data frame de 5 variables más la variable clase, donde todas las variables son de tipo entero. El dataset contiene 150 filas. Con **summary** podemos hacernos una primera idea de cómo es el dataset: por ejemplo, vemos que la variable X1 es una variable categórica que solo toma valores 1 y 2, que la variable X23 toma valores entre 1 y 25 y que parece estar bien distribuida. La variable X3 toma valores entre 1 y 26, X1.1 es otra varible categórica que toma valores entre 1 y 2 y X19 toma valores entre 3 y 66. Por último, la clase (X3.1) toma valores entre 1 y 3, y parece estar bien distribuida, por lo que seguramente el número de muestras de cada clase esté balanceado. No hay diferencias entre medias y medianas llamativas, por lo que no parece a primera vista que ninguna variable presente algún sesgo importante.

Pasamos a calcular medidas de dispersión como la desviación estándar(SD), desviación absoluta media(MAD), asimetría y curtosis y el rango intercuartílico (IQR), aunque antes, de nuevo comprobamos si hay valores perdidos en el dataset o repetidos:

```
sum(which(is.na(tae)))
```

```
## [1] 0
```

```
duplicated(tae)
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [34] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [45]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [56]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [67]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##  [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [111] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [122] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [144] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

El dataset no tiene ningún valor perdido. Calculamos la media, mediana y desviación típica de cada variable:

```
apply(tae,2,mean)
```

```
##        X1       X23        X3      X1.1       X19      X3.1
##  1.813333 13.580000  8.140000  1.853333 27.926667  2.013333
```

```
apply(tae,2,median)
```

```
##   X1  X23   X3 X1.1  X19 X3.1
##  2.0 13.0  4.5  2.0 27.0  2.0
```

```
apply(tae,2,sd)
```

```
##         X1        X23         X3       X1.1        X19       X3.1
##  0.3909491  6.8053177  7.0349368  0.3549585 12.9164048  0.8191227
```

Vemos que hay variables que será interesante analizar con más detalle, como X19, ya que a pesar de que su media y mediana son casi idénticas, presenta una desviación típica más alta que ninguna otra variable. También merece análisis la variable X3, ya que su media y mediana son muy distantes, y su desviación típica está en torno a 7, lo cual indica que los datos presentan alguna anomalía o variabilidad en su distribución. Calculamos la MAD (median absolute deviation), que calcula la media de las desviaciones absolutas a la mediana:

```
apply(tae,2,mad)
```

```
##      X1     X23      X3    X1.1     X19    X3.1
##  0.0000  8.8956  4.4478  0.0000 14.0847  1.4826
```

Vemos que las variables X1 y X1.1 no presentan desviación absoluta con respecto a su mediana, al contrario del resto de variables. La variable clase (X3.1) sí presenta desviación con respecto a su mediana, aunque no muy grane. De nuevo, resulta llamativo el alto valor MAD de la variable X19. Calculamos el rango intercuartil (IQR) de las variables, por ser la medida de variabilidad más robusta.Se calcula a partir de los cuartiles 1º y 3º:

```
apply(tae,2,quantile)
```

```
##      X1 X23   X3 X1.1 X19 X3.1
## 0%    1   1  1.0    1   3    1
## 25%   2   8  3.0    2  19    1
## 50%   2  13  4.5    2  27    2
## 75%   2  20 15.0    2  37    3
```

```
## 100%  2  25 26.0    2  66    3
```

```
apply(tae,2,IQR)
```

```
##   X1  X23   X3 X1.1  X19 X3.1
##    0   12   12    0   18    2
```

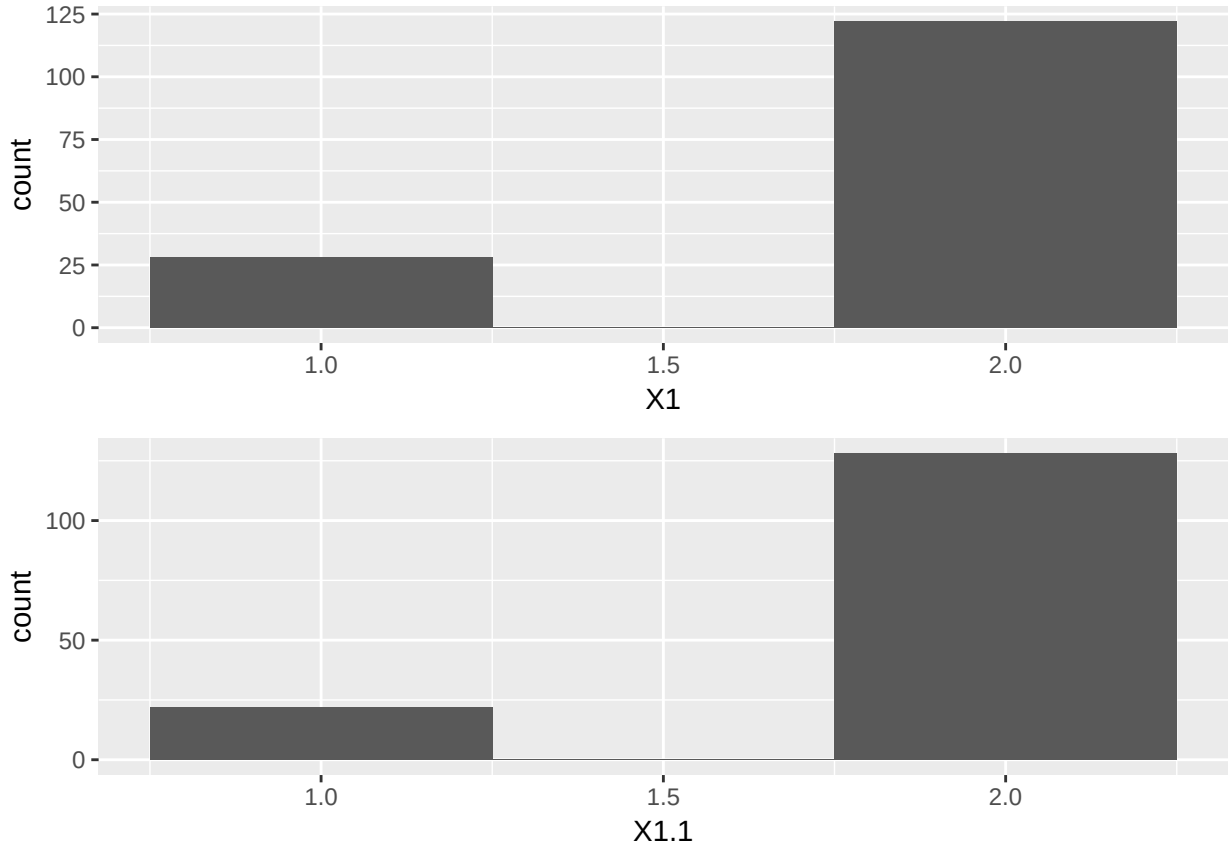
Con *quantile* calculamos los tres cuartiles de cada variable. El IQR define el rango en el que se encuentran el 50% de los valores, por lo que se dice que es una medida muy robusta. Observamos que las variables X19, X3 y X23 presentan valores de IQR más altos y por lo tanto variabilidad en los datos. Podemos saber más sobre el tipo de variabilidad que presentan calculando el *skewness* y *curtosis*, que nos darán información de cómo de asimétricas y picudas son las distribuciones:

```
library(moments)
skewness(tae)
```

```
##           X1          X23           X3         X1.1          X19
## -1.608306890  0.001726227  0.851239237 -1.997512658  0.482458615
##         X3.1
## -0.024495986
```

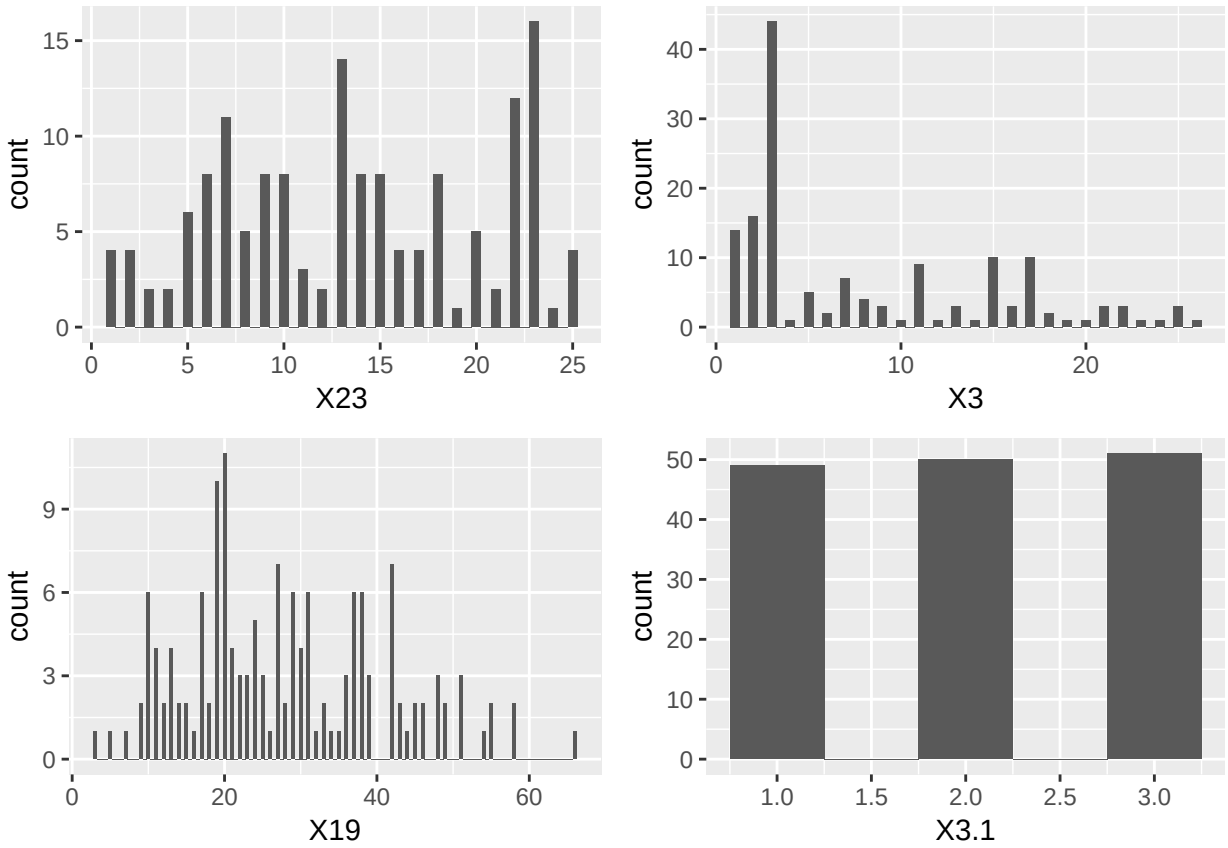
Como mencionamos anteriormente, valores de *skewness* positivos o negativos lejanos a 0 indican un sesgo o desplazamiento en la distribución. En este dataset la mayoría de las variables tienen un valor de skewness cercano a 0, por lo que no presentan desplazamientos grandes o importantes. Las variables X1.1 y X1 sí presentan un pequeño sesgo negativo (los datos se concentran a la derecha de la distribución):

```
library(ggplot2)
par(mfrow=c(1,2))
plot1<-ggplot(tae, aes(X1))+geom_histogram(binwidth = 0.5)
plot2<-ggplot(tae, aes(X1.1))+geom_histogram(binwidth = 0.5)
gridExtra::grid.arrange(plot1,plot2)
```



que si los representamos, recordamos que X1 y X1.1 eran variables categóricas y por tanto, ese "desplamiento" se debe a que hay más valores en la 2º categoría que en la 1º.

```
library(ggplot2)
par(mfrow=c(1,2))
plot1<-ggplot(tae, aes(X23))+geom_histogram(binwidth = 0.5)
plot2<-ggplot(tae, aes(X3))+geom_histogram(binwidth = 0.5)
plot3<-ggplot(tae, aes(X19))+geom_histogram(binwidth = 0.5)
plot4<-ggplot(tae, aes(X3.1))+geom_histogram(binwidth = 0.5)
gridExtra::grid.arrange(plot1,plot2,plot3,plot4)
```



Del resto de variables podemos decir que X3 y X19 presentan un pequeño sesgo positivo, tal y como se aprecia en sus histogramas, X23 presenta un pequeño sesgo negativo y X3.1 (la variable clase) está bastante bien distribuida. Esto último lo podemos ilustrar también con una tabla de contingencia:

```
table(tae$X3.1)
```

```
##
##  1  2  3
## 49 50 51
```

Donde vemos que la dos primeras clases contienen 36 muestras y la tercera clase contiene 38. Por último, podemos calcular la curtosis de la distribución, que nos indicará como de picuda es la distribución en el centro:

```
kurtosis(tae)
```

```
##       X1      X23       X3     X1.1      X19     X3.1
## 3.586651 1.854475 2.458203 4.990057 2.613212 1.500800
```

Vemos que el mayor pico lo presenta X1.1 donde se concentran los datos de la segunda categoría. También contiene un pico X1 por la misma razón. X3 presenta un pico en torno al valor 3 y X19 en torno al valor 20.

Visualización

Lo primero que vamos a hacer es cambiar los nombres de las variables, para hacer más intuitiva su representación:

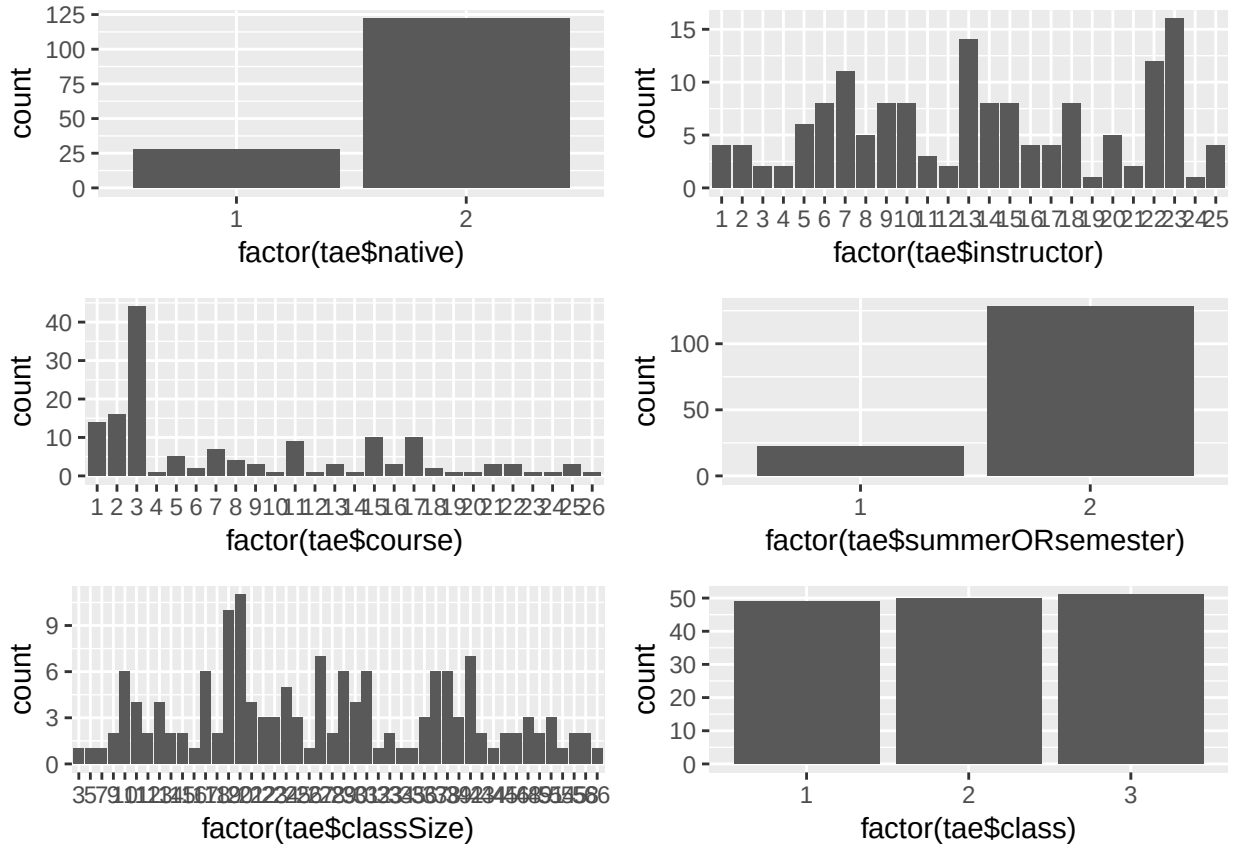
```
names(tae)
```

```
## [1] "X1"   "X23"  "X3"   "X1.1" "X19"  "X3.1"
```

```
names(tae)<-c("native","instructor","course","summerORsemester","classSize","class")
```

Una vez cambiados los nombres, representamos individualmente cada una de las 6 variables:

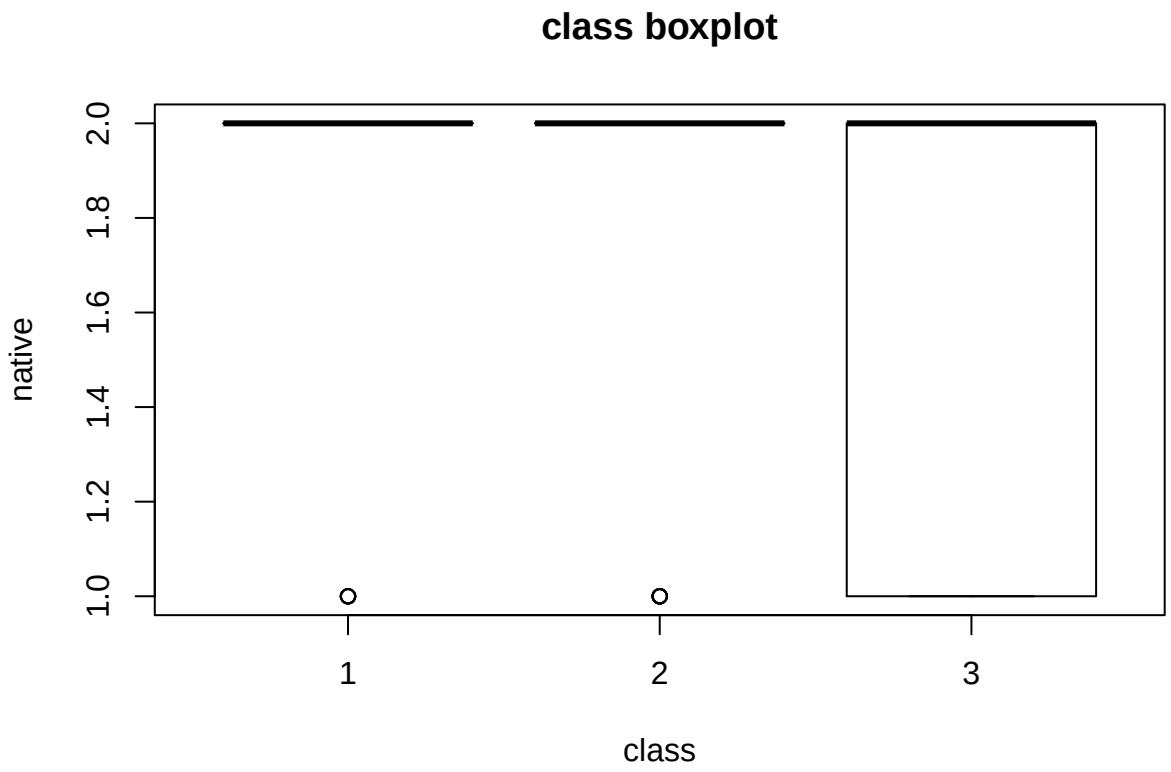
```
p1<-qplot(factor(tae$native))
p2<-qplot(factor(tae$instructor))
p3<-qplot(factor(tae$course))
p4<-qplot(factor(tae$summerORsemester))
p5<-qplot(factor(tae$classSize))
p6<-qplot(factor(tae$class))
gridExtra::grid.arrange(p1,p2,p3,p4,p5,p6)
```



Vemos que todas son variables nominales que quizás sería conveniente "dumizar" para convertirlas en variables indicativas antes de ajustar el modelo de clasificación, o normalizarlas, por ejemplo "native", "summerORsemester" o "classSize".

Podemos ver cómo se distribuye una variable con respecto a otra, por ejemplo, la clase frente a si el profesor es nativo:

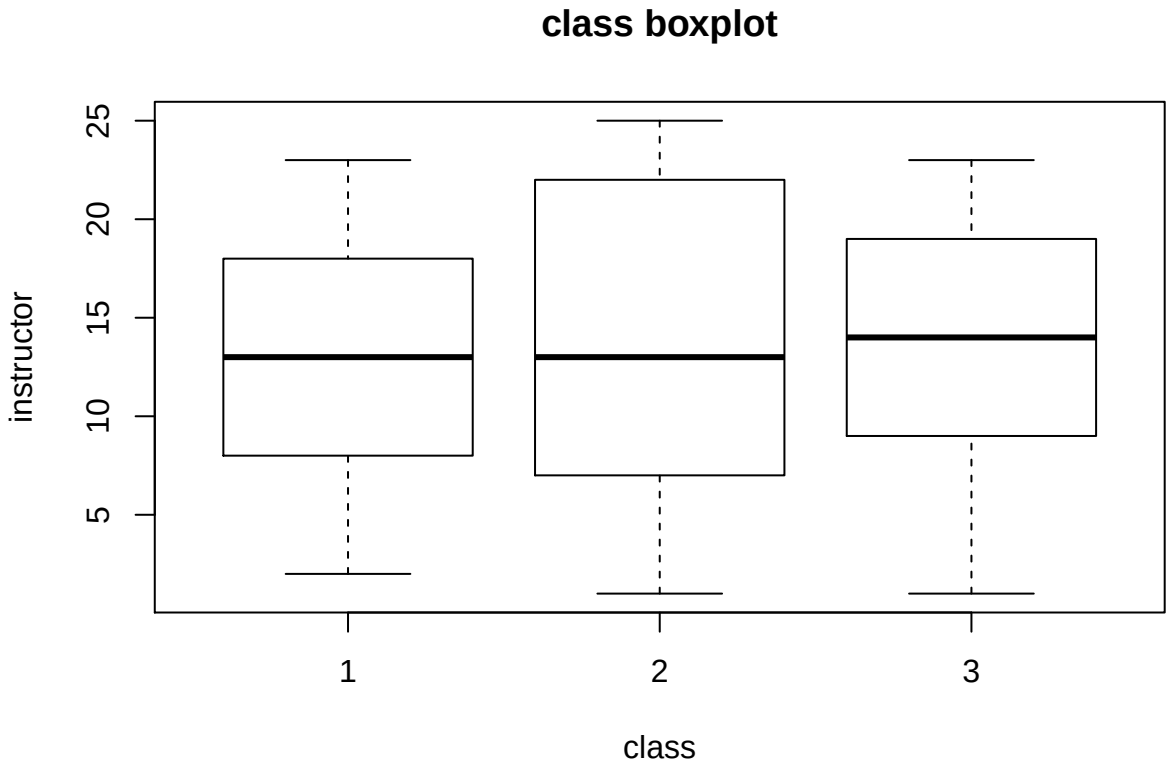
```
boxplot(native~class,data=tae,xlab="class",ylab="native",main="class boxplot")
```



Vemos que todos los profesores nativos están en la clase 3, mientras que en las clases 1 y 2 no hay profesores asistentes que sean nativos (que tengan inglés como lengua materna).

Si representamos los instructores del curso frente a la clase:

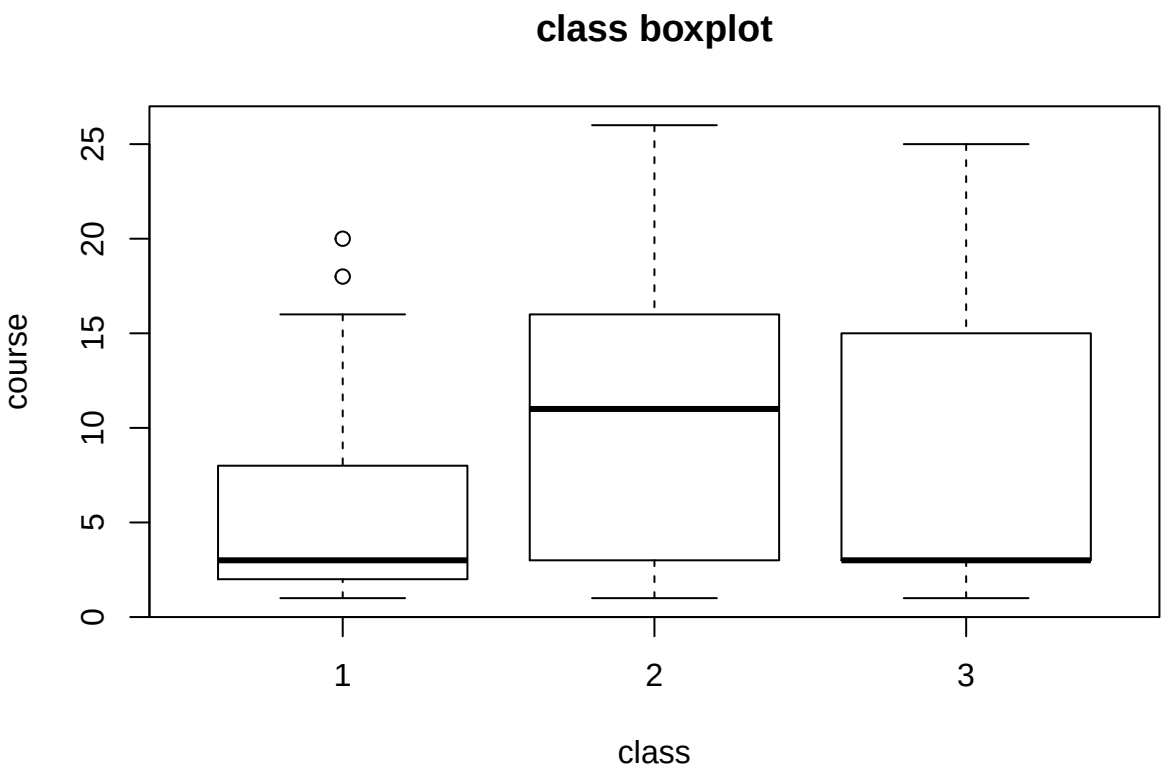
```
boxplot(instructor~class,data=tae,xlab="class",ylab="instructor",main="class boxplot")
```



No estoy segura de lo que la variable “instructor” representa, pero si representa el “identificador” del profesor que impartió dicho curso, no creo que nos pueda servir de mucho en nuestro modelo, si bien es cierto que no todos los instructores tienen el mismo número de evaluaciones recogidas en el dataset, por ejemplo “1” aparece 3 veces y “13” aparece 10 veces.

Si representamos el curso frente a la clase:

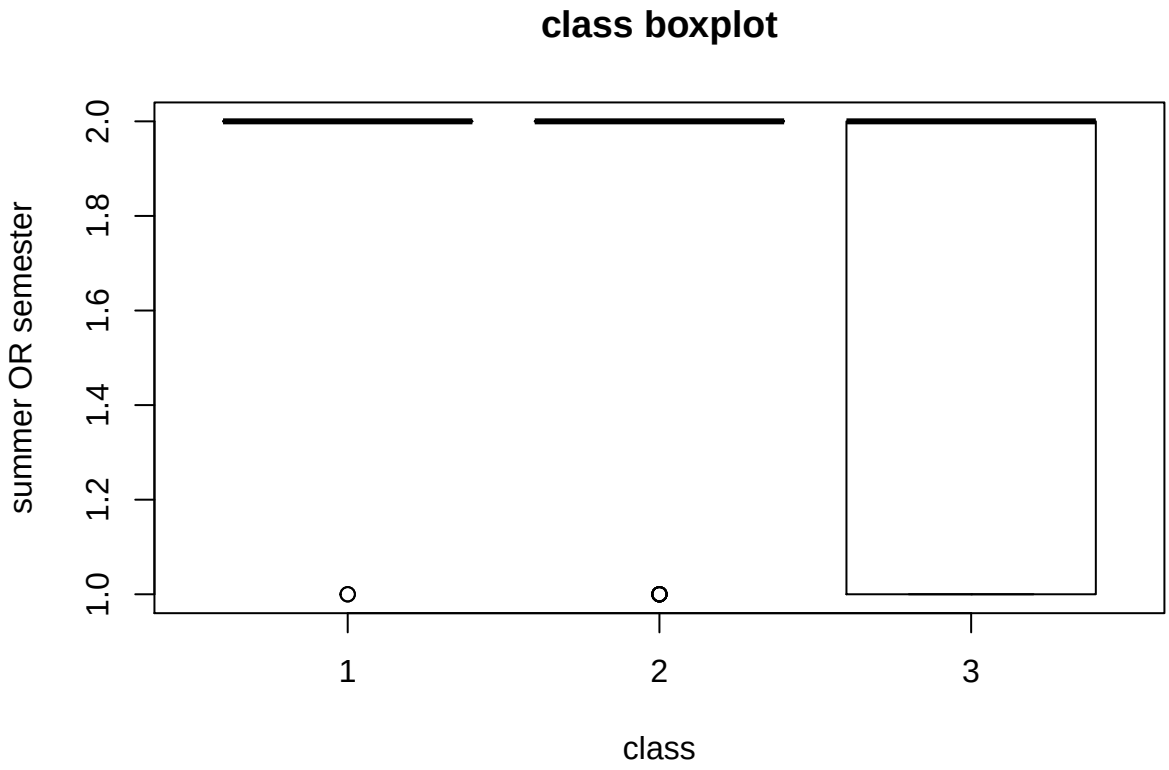
```
boxplot(course~class,data=tae,xlab="class",ylab="course",main="class boxplot")
```



Recordamos que la variable “class” denota la puntuación obtenida por esos profesores, que puede ser baja, media o alta (1,2,3 respectivamente). Podemos observar en el gráfico que las puntuaciones bajas (clase 1) vienen de cursos enumerados entre 1 y 8, aunque algunos de estos valores se solapan con los de las clases 2 y 3. Dicho de otra manera, profesores de cursos con numeración inferior a 2.5 obtienen mala calificación. Profesores de cursos con numeración entre 2.5 y 7.5 pueden obtener calificación “Low”(clase 1), “medium”(clase 2) o “high”(clase 3), profesores de cursos con numeración entre 7.5 y 15 obtienen calificación “medium” o “high” y profesores de cursos con numeración superior a 15 obtienen de calificación “medium”(clase 2), a excepción de tres outliers que pertenecen a la clase 1 y que tienen valores de curso superior a 15. Parece que esta variable aporta bastante información a la salida.

Si analizamos si se impartió en verano o durante un semestre, frente a la clase:

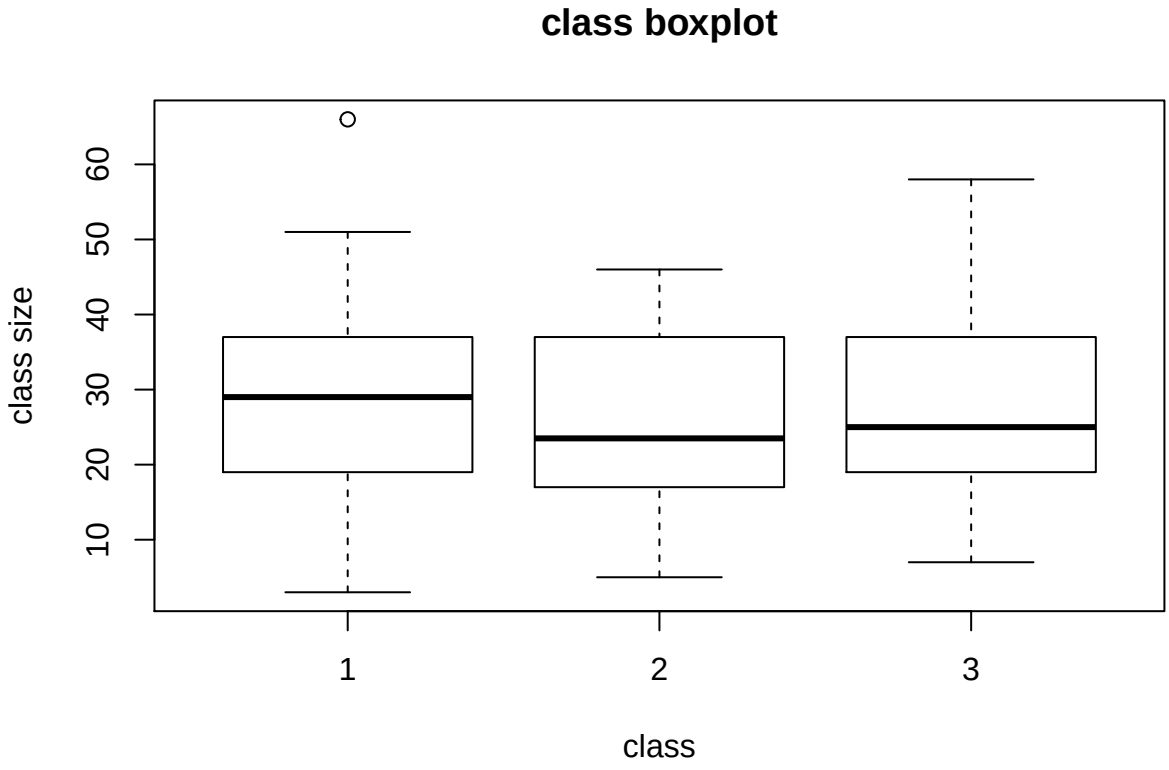
```
boxplot(summerORsemester~class,data=tae,xlab="class",ylab="summer OR semester",main="class boxplot")
```



Observamos que ocurre algo parecido a lo que ocurría con la variable "native". Tenemos que los profesores que obtuvieron una puntuación de "high" (clase 3) absolutamente todos dieron clase durante semestres regulares, mientras que los que obtuvieron "low" (clase 1) o "medium"(clase 2), dieron clase durante el verano. Parece que dar clase durante un sememestre regular y ser nativo son motivo de una puntuación "high" (clase 3). ¿Pero qué hace que se obtenga una puntuación "medium" o "low" además de no ser nativo y dar clase solo en cursos de verano? Seguimos analizando variables.

La última variable que nos queda por analizar es el tamaño de la clase frente a la variable clase:

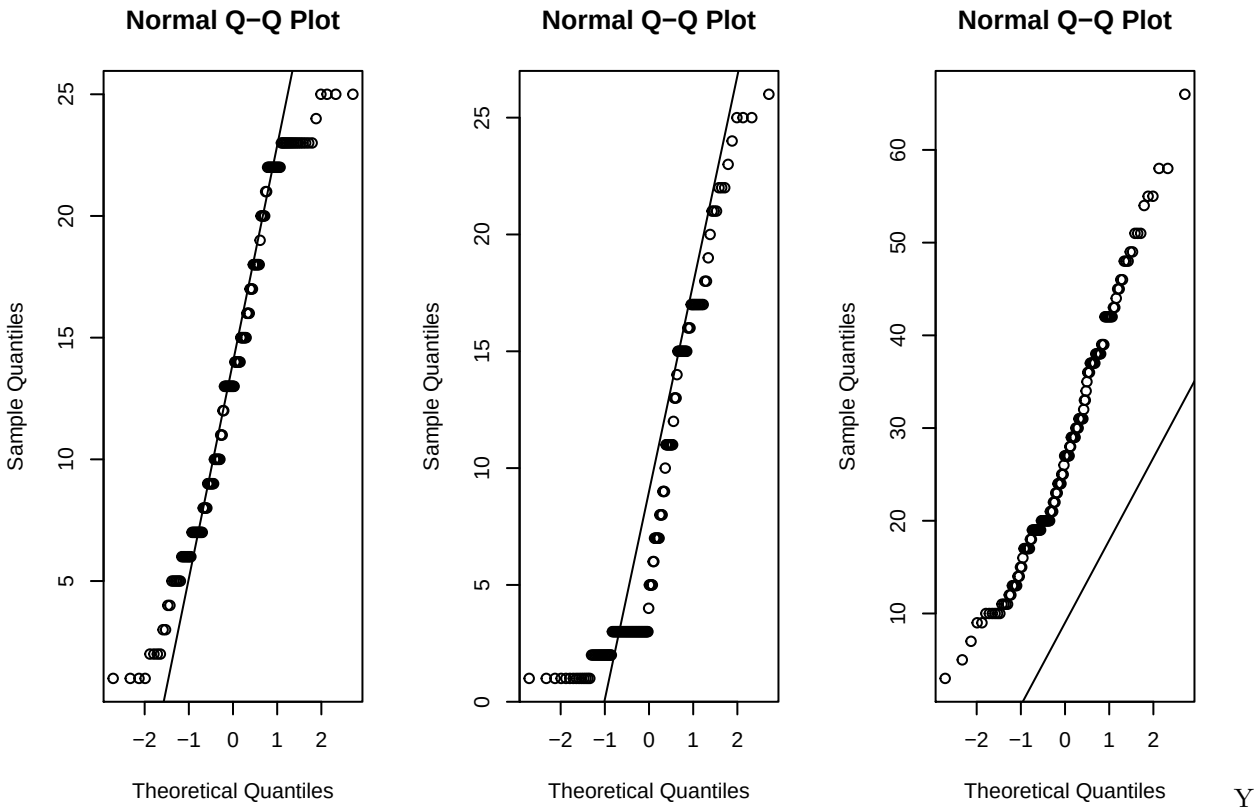
```
boxplot(classSize~class,data=tae,xlab="class",ylab="class size",main="class boxplot")
```



Vemos que el tamaño de la clase no es un atributo que discrimine mucho por sí solo, pero sí que se aprecia en el boxplot que en clases de menos de 20 estudiantes, la nota del profesor es "medium" (clase 2). También podemos ver que para la clase 1 (puntuación "low"), la media de alumnos por clase a sido de ~30 alumnos, aunque hay un outlier de más de 60 alumnos por clase. Para la clase 2 (puntuación "medium"), la media de alumnos por clase ha sido de ~23, si bien la mayoría de las clases donde el profesor o profesora obtuvo "medium" tenía entre 23 y 38 alumnos por clase. Para la clase 3 (puntuación "high") la media de alumnos por clase era de 25, si bien la mayoría de los profesores con puntuación "high" dió clase en aulas con entre 25 y 38 alumnos.

Por último, para asegurarnos de si los datos siguen una distribución normal o cercana a ella, hacemos QQ-plots de las variables nominales:

```
par(mfrow=c(1,3))
qqnorm(tae$instructor)
qqline(tae$instructor)
qqnorm(tae$course)
qqline(tae$course)
qqnorm(tae$classSize)
qqline(tae$course)
```



Y observamos que la variable instructor parece que sí sigue una distribución normal, pero no la variable "course" o "classSize", que habría que normalizarlas antes de aplicar el modelo. Las variables categóricas "native", "summerORsemester" y "class" no las representamos por ser categóricas.

Apartado Clasificación

k-NN con diferentes valores de k

En primer lugar aplicamos un pequeño preprocesamiento. Las variables "native" y "summerORSemester" que tomaban valor 1 o 2, las cambiamos para que tomen valor 1 o 0, y convertimos en variables dummy el resto de variables nominales, convirtiendolas en variables indicativas. Por último, convertimos la variable "class" en factores para poder aplicar KNN:

```
taeKnn<-tae
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
taeKnn$native<-ifelse(taeKnn$native==2,0,1)
taeKnn$summerORsemester<-ifelse(taeKnn$summerORsemester==2,0,1)
taeKnn$course<-dummy(taeKnn$course)
taeKnn$classSize<-dummy(taeKnn$classSize)
taeKnn$instructor<-dummy(taeKnn$instructor)
taeKnn$class<-as.factor(taeKnn$class)
```

Creamos conjuntos de train y test y sus respectivas etiquetas:

```
shuffle_ds <- sample(dim(taeKnn)[1])
traindat <- (dim(taeKnn)[1] * 80) %/% 100
tae_train <- taeKnn[shuffle_ds[1:traindat], ]
```

```
tae_test <- taeKnn[shuffle_ds[(traindat+1):dim(taeKnn)[1]], ]

tae_train_labels <- taeKnn[shuffle_ds[1:traindat], ncol(taeKnn)]
tae_test_labels <- taeKnn[shuffle_ds[(traindat+1):dim(taeKnn)[1]], ncol(taeKnn)]
```

Entrenamos el modelo usando **caret**, especificando que queremos que nos encuentre el mejor k entre 1 y 20. Para que el modelo sea replicable, fijamos una semilla:

```
require(caret)
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:kknn':
##
##     contr.dummy
```

```
set.seed(6)
knnModel <- train(x = tae_train, y = tae_train_labels, method = "knn",metric="Accuracy", tuneGrid = data
class(knnModel)
```

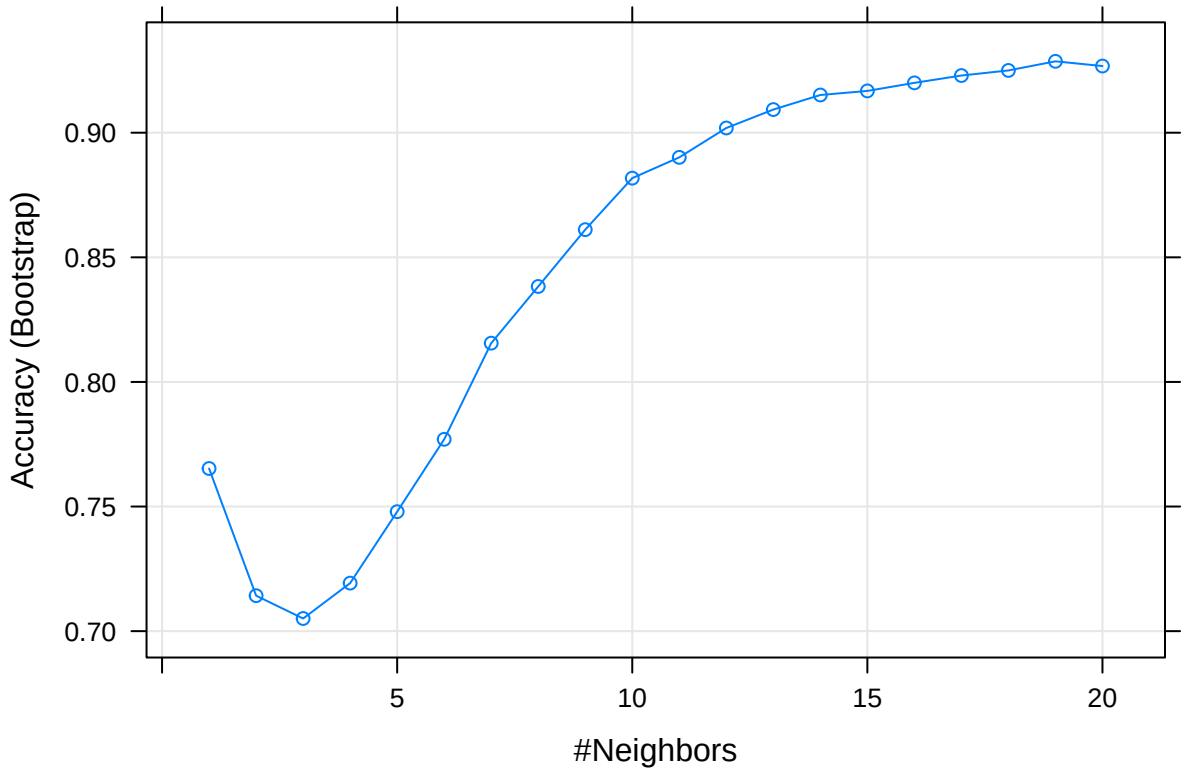
```
## [1] "train"
```

```
knnModel
```

```
## k-Nearest Neighbors
##
## 120 samples
##   6 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 120, 120, 120, 120, 120, 120, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.7652638  0.6469623
##    2  0.7142250  0.5694866
##    3  0.7050685  0.5565984
##    4  0.7192741  0.5785517
##    5  0.7479346  0.6211415
##    6  0.7769729  0.6642485
##    7  0.8155492  0.7219905
##    8  0.8383010  0.7555450
##    9  0.8611106  0.7898281
##   10  0.8817678  0.8205016
##   11  0.8901142  0.8331323
##   12  0.9018982  0.8508697
##   13  0.9092896  0.8621765
##   14  0.9151408  0.8709761
##   15  0.9167735  0.8736311
##   16  0.9200053  0.8783979
##   17  0.9229446  0.8827170
```

```
##   18  0.9249866  0.8859401
##   19  0.9286472  0.8913637
##   20  0.9267611  0.8886610
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was k = 19.
```

```
plot(knnModel)
```



Vemos que el mejor modelo lo obtiene con k=14, con un accuracy de 0.9050387 y un valor kappa de 0.8562562, por lo que parece que es un modelo bastante bueno. Para evaluarlo, calculamos el error en test usando caret y de forma manual, para comprobar que es correcto:

```
knnPred <- predict(knnModel, newdata =tae_test)
knnPred
```

```
##  [1] 3 2 3 1 1 1 3 3 2 2 2 2 2 1 2 1 2 2 2 1 1 2 2 3 3 3 3 1 3 3
## Levels: 1 2 3
```

```
postResample(pred = knnPred, obs = taeKnn[shuffle_ds[(traindat+1):dim(taeKnn)[1]], ncol(taeKnn)])
```

```
##  Accuracy     Kappa
## 0.9333333 0.8979592
```

```
prediction<-predict(knnModel,tae_test)
d<-table(prediction,tae_test_labels)
sum(diag(d))/sum(d)
```

```
## [1] 0.9333333
```

```
1-sum(diag(d))/sum(d)
```

```
## [1] 0.06666667
```

Vemos que obtenemos un accuracy en test de 0.9666667 y un error en test de (1-0.9666667)=0.03333, mejorando el resultado que hay anotado en la tabla de *clasif_test_alumnos.csv* para KNN, siendo en la tabla el error de 0.383809523809524. También mejora el error de train, que es de (1-0.9532644)=0.0467356 frente a 0.526346047540077 que hay anotado en la tabla de *clasif_train_alumnos.csv*.

LDA

Como cada algoritmo funciona de forma diferente, por ejemplo LDA necesita de datos normalizados y con clases convarianzas similares, el preprocesamiento aplicado en el paso anterior funciona muy bien para KNN,pero quizás no tan bien para LDA, dado que, tal como están los datos, solo una variable tiene varianza distinta de 0. Por tanto, les aplicamos otro preprocesamiento más acorde al algoritmo al que nos enfrentamos:

Comenzamos calculando la varianza de las variables:

```
apply(tae,2,var)
```

```
##           native       instructor           course summerORsemester
##        0.1528412       46.3123490       49.4903356        0.1259955
##        classSize            class
##      166.8335123        0.6709620
```

Vemos que tenemos dos variables predictoras ("native" y "summerORsemester") con varianzas cercanas a 0, sin embargo, en principio no las vamos a eliminar porque son variables que aportan mucha información. Lo siguiente que haremos es calcular la matriz de correlaciones:

```
cor(tae)
```

```
##                      native  instructor     course summerORsemester
## native            1.0000000 -0.23651722  0.1266975        0.1882945
## instructor       -0.2365172  1.00000000 -0.2311922       -0.1506980
## course            0.1266975 -0.23119220  1.0000000        0.2179163
## summerORsemester  0.1882945 -0.15069799  0.2179163        1.0000000
## classSize        -0.1635480 -0.02967214 -0.0369642        0.2640578
## class            -0.2436687  0.06482185  0.1499169       -0.2702219
##                    classSize       class
## native          -0.16354802 -0.24366875
## instructor      -0.02967214  0.06482185
## course          -0.03696420  0.14991691
## summerORsemester  0.26405777 -0.27022185
## classSize         1.00000000 -0.03035536
## class            -0.03035536  1.00000000
```

Vemos que las mayores correlaciones positivas que presenta la clase son con las variables "native" y "summerORsemester", aunque juzgar por el valor de correlación no parece una correlación muy fuerte. Se aprecia una correlación negativa entre "summer OR semester" y "course" y otra correlación positiva entre "summer OR semester" y "classSize". También parecen correladas "native" e "instructor", aunque en ningún caso los valores de correlación con elevados. Para constrastar los resultados, calculamos la covarianza:

```
cov(tae)
```

```
##                       native  instructor      course summerORsemester
## native            0.15284116  -0.6292617   0.3484564       0.02612975
## instructor       -0.62926174  46.3123490 -11.0683221      -0.36402685
## course            0.34845638 -11.0683221  49.4903356       0.54416107
## summerORsemester  0.02612975  -0.3640268   0.5441611       0.12599553
## classSize        -0.82586130  -2.6081879  -3.3587919       1.21064877
## class            -0.07803132   0.3613423   0.8638926      -0.07856823
```

```
##                     classSize       class
## native            -0.8258613 -0.07803132
## instructor        -2.6081879  0.36134228
## course            -3.3587919  0.86389262
## summerORsemester   1.2106488 -0.07856823
## classSize        166.8335123 -0.32116331
## class             -0.3211633  0.67096197
```

Podemos ver que "course" e "instructor" son los únicos que presentan una covarianza alta y, en este caso, negativa de -11.0683221. Puesto que "course" además está relacionada con "classSize" aunque no demasiado, eliminaremos "course" del modelo. En el resto de variables no se aprecian dependencias significativas.

Como LDA asume que las variables siguen una distribución normal, las normalizamos como paso previo al ajuste:

```
taeLda<-tae
taeLda$native<-scale(taeLda$native,center = TRUE)
taeLda$instructor<-scale(taeLda$instructor,center = TRUE)
taeLda$summerORsemester<-scale(taeLda$summerORsemester,center = TRUE)
taeLda$classSize<-scale(taeLda$classSize,center = TRUE)
taeLda$class<-as.factor(taeLda$class)
```

```
shuffle_ds <- sample(dim(taeLda)[1])
traindat <- (dim(taeLda)[1] * 80) %/% 100
tae_train <- taeLda[shuffle_ds[1:traindat], ]
tae_test <- taeLda[shuffle_ds[(traindat+1):dim(taeLda)[1]], ]

tae_train_labels <- taeLda[shuffle_ds[1:traindat], ncol(taeLda)]
tae_test_labels <- taeLda[shuffle_ds[(traindat+1):dim(taeLda)[1]], ncol(taeLda)]
```

```
library(MASS)
library(ISLR)
set.seed(2)
lda.fit<-lda(class~native+instructor+summerORsemester+classSize,data=tae_train)
lda.fit
```

```
## Call:
## lda(class ~ native + instructor + summerORsemester + classSize,
##     data = tae_train)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3333333 0.3000000 0.3666667
##
## Group means:
##        native  instructor summerORsemester   classSize
## 1  0.28562971 -0.16237302        0.2723323  0.15277729
## 2  0.05115756 -0.01991906        0.0219118 -0.14486315
## 3 -0.27826613  0.07841461       -0.3551418  0.05318583
##
## Coefficients of linear discriminants:
##                           LD1        LD2
## native            0.62526618  0.1843847
## instructor       -0.02732948 -0.1565465
## summerORsemester  0.69563792 -0.3253620
## classSize        -0.04216639  1.0605376
##
```

```
## Proportion of trace:
##    LD1    LD2
## 0.8715 0.1285
```

```
lda.pred<-predict(lda.fit,tae_test)
mean(lda.pred$class==tae_test$class)
```

```
## [1] 0.4333333
```

```
d<-table(lda.pred$class,tae_test_labels)
sum(diag(d))/sum(d)
```

```
## [1] 0.4333333
```

```
1-sum(diag(d))/sum(d)
```

```
## [1] 0.5666667
```

QDA

Utilizando los mismos datos de antes y los mismos conjuntos de train y test:

```
library(MASS)
library(ISLR)
set.seed(2)
qda.fit<-qda(class~native+instructor+summerORsemester+classSize,data=tae_train)
lda.fit
```

```
## Call:
## lda(class ~ native + instructor + summerORsemester + classSize,
##     data = tae_train)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3333333 0.3000000 0.3666667
##
## Group means:
##        native  instructor summerORsemester   classSize
## 1  0.28562971 -0.16237302        0.2723323  0.15277729
## 2  0.05115756 -0.01991906        0.0219118 -0.14486315
## 3 -0.27826613  0.07841461       -0.3551418  0.05318583
##
## Coefficients of linear discriminants:
##                           LD1        LD2
## native             0.62526618  0.1843847
## instructor        -0.02732948 -0.1565465
## summerORsemester   0.69563792 -0.3253620
## classSize         -0.04216639  1.0605376
##
## Proportion of trace:
##    LD1    LD2
## 0.8715 0.1285
```

```
qda.pred<-predict(qda.fit,tae_test)
mean(qda.pred$class==tae_test$class)
```

```
## [1] 0.4
```

```r
d<-table(qda.pred$class,tae_test_labels)
sum(diag(d))/sum(d)
```

```
## [1] 0.4
```

```r
1-sum(diag(d))/sum(d)
```

```
## [1] 0.6
```

Comparar los resultados de los tres algoritmos

Puesto que antes no hemos validado los resultados de los modelos con varias particiones, vamos a hacer uso de caret para hacer 10 particiones de los datos para hacer CV. Como hemos guardado una copia de cada dataset preprocesado para cada algoritmo, los podemos usar directamente. Empezamos haciendo validación cruzada con 10 particiones para el modelo de KNN:

```r
taeKnntrainIndex <- createDataPartition(taeKnn$class, p = .8,
                                  list = F,
                                  times = 1)
taeKnnTrain <- taeKnn[taeKnntrainIndex, ]
taeKnnTest <- taeKnn[-taeKnntrainIndex, ]
table(taeKnnTrain$class)
```

```
##
##  1  2  3
## 40 40 41
```

```r
fitControl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 10)
knnfit <- train(x=taeKnnTrain,y=taeKnnTrain$class,
                  method = "knn",
                  trControl = fitControl,
                  tuneGrid = data.frame(.k=1:20),
                  metric = "Accuracy")
knnfit
```

```
## k-Nearest Neighbors
##
## 121 samples
##   6 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 109, 109, 108, 109, 109, 109, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.7752564  0.6628181
##    2  0.7263462  0.5893868
##    3  0.7523077  0.6284276
##    4  0.7623718  0.6434820
##    5  0.7878205  0.6816899
##    6  0.8322436  0.7484192
```

```
##    7  0.8801923  0.8202402
##    8  0.8819231  0.8227664
##    9  0.9024359  0.8535703
##   10  0.9048718  0.8572006
##   11  0.9149359  0.8722899
##   12  0.9256410  0.8883508
##   13  0.9371795  0.9056827
##   14  0.9437179  0.9154726
##   15  0.9452564  0.9178362
##   16  0.9445513  0.9167015
##   17  0.9528846  0.9292015
##   18  0.9560897  0.9340758
##   19  0.9560897  0.9340545
##   20  0.9585256  0.9377365
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was k = 20.
```

```
prediction <- predict(knnfit, taeKnnTest)
TestAccuracy <- postResample(prediction,taeKnnTest$class)
TestAccuracy
```

```
##  Accuracy     Kappa
## 0.9655172 0.9482143
```

Hacemos lo mismo para LDA:

```
taeLdatrainIndex <- createDataPartition(taeLda$class, p = .8,
                                  list = F,
                                  times = 1)
taeLdaTrain <- taeLda[taeLdatrainIndex, ]
taeLdaTest <- taeLda[-taeLdatrainIndex, ]
table(taeLdaTrain$class)
```

```
##
##  1  2  3
## 40 40 41
```

```
fitControl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 10)
ldafit <- train(class~.,taeLdaTrain,method="lda",trControl = fitControl,metric="Accuracy")

ldafit
```

```
## Linear Discriminant Analysis
##
## 121 samples
##   5 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 109, 109, 109, 109, 109, 109, ...
## Resampling results:
##
##   Accuracy   Kappa
```

```
##   0.4970513  0.2463859
```

```
prediction <- predict(ldafit, taeLdaTest)
TestAccuracy <- postResample(prediction,taeLdaTest$class)
TestAccuracy
```

```
##  Accuracy     Kappa
## 0.5517241 0.3327434
```

Y para QDA:

```
taeLdatrainIndex <- createDataPartition(taeLda$class, p = .8,
                                  list = F,
                                  times = 1)
taeLdaTrain <- taeLda[taeLdatrainIndex, ]
taeLdaTest <- taeLda[-taeLdatrainIndex, ]
table(taeLdaTrain$class)
```

```
##
##  1  2  3
## 40 40 41
```

```
fitControl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 10)
qdafit <- train(class~.,taeLdaTrain,method="qda",trControl = fitControl,metric="Accuracy")
```

```
## Warning: model fit failed for Fold09.Rep08: parameter=none Error in qda.default(x, grouping, ...) : i
```

```
## Warning: model fit failed for Fold02.Rep10: parameter=none Error in qda.default(x, grouping, ...) : i
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
qdafit
```

```
## Quadratic Discriminant Analysis
##
## 121 samples
##   5 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 109, 109, 109, 109, 108, 109, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.4727891  0.2094491
```

```
prediction <- predict(qdafit, taeLdaTest)
TestAccuracy <- postResample(prediction,taeLdaTest$class)
TestAccuracy
```

```
##  Accuracy     Kappa
## 0.6896552 0.5388693
```

Como queremos añadir los resultados a las tablas para copararlos de forma general, no hemos especificado variables predictoras en lda y qda , a diferencia de como hicimos arriba. Ante los datos que tenemos, podemos concluir que en media en las 10 particiones, se han obtenido mejores resultados con KNN para este dataset,

quizás por su alto contenido de variables categóricas y nominales, mientras que entre LDA y QDA no se aprecia gran diferencia en calidad.