FEST: Relación ejercicios Tema 3-5

Alumno: Inmaculada Perea Fernández

Enero 2017

1 Obtención del conjunto de datos para el estudio

Carga de los paquetes necesarios, si no los tiene instalados utilice el comando install.packages

```
library(AppliedPredictiveModeling)
library(MASS)
library(car)
library(ISLR)
library(leaps)
library(caret)
library(ggplot2)
library(scatterplot3d)
```

Carga de los ficheros de datos: solTrainX, solTraintransX, solTrainY, solTestX, solTestXtrans, solTestY
Visualiza las variables de los ficheros

```
data(solubility)
ls(pattern = "^solT")
```

```
## [1] "solTestX"       "solTestXtrans"  "solTestY"       "solTrainX"
## [5] "solTrainXtrans" "solTrainY"
```

```
# Fichero solTrainX
names(solTrainX)
```

```
##   [1] "FP001"             "FP002"             "FP003"
##   [4] "FP004"             "FP005"             "FP006"
##   [7] "FP007"             "FP008"             "FP009"
##  [10] "FP010"             "FP011"             "FP012"
##  [13] "FP013"             "FP014"             "FP015"
##  [16] "FP016"             "FP017"             "FP018"
##  [19] "FP019"             "FP020"             "FP021"
##  [22] "FP022"             "FP023"             "FP024"
##  [25] "FP025"             "FP026"             "FP027"
##  [28] "FP028"             "FP029"             "FP030"
##  [31] "FP031"             "FP032"             "FP033"
##  [34] "FP034"             "FP035"             "FP036"
##  [37] "FP037"             "FP038"             "FP039"
##  [40] "FP040"             "FP041"             "FP042"
##  [43] "FP043"             "FP044"             "FP045"
##  [46] "FP046"             "FP047"             "FP048"
##  [49] "FP049"             "FP050"             "FP051"
##  [52] "FP052"             "FP053"             "FP054"
##  [55] "FP055"             "FP056"             "FP057"
##  [58] "FP058"             "FP059"             "FP060"
##  [61] "FP061"             "FP062"             "FP063"
##  [64] "FP064"             "FP065"             "FP066"
##  [67] "FP067"             "FP068"             "FP069"
```

```
##  [70] "FP070"             "FP071"             "FP072"
##  [73] "FP073"             "FP074"             "FP075"
##  [76] "FP076"             "FP077"             "FP078"
##  [79] "FP079"             "FP080"             "FP081"
##  [82] "FP082"             "FP083"             "FP084"
##  [85] "FP085"             "FP086"             "FP087"
##  [88] "FP088"             "FP089"             "FP090"
##  [91] "FP091"             "FP092"             "FP093"
##  [94] "FP094"             "FP095"             "FP096"
##  [97] "FP097"             "FP098"             "FP099"
## [100] "FP100"             "FP101"             "FP102"
## [103] "FP103"             "FP104"             "FP105"
## [106] "FP106"             "FP107"             "FP108"
## [109] "FP109"             "FP110"             "FP111"
## [112] "FP112"             "FP113"             "FP114"
## [115] "FP115"             "FP116"             "FP117"
## [118] "FP118"             "FP119"             "FP120"
## [121] "FP121"             "FP122"             "FP123"
## [124] "FP124"             "FP125"             "FP126"
## [127] "FP127"             "FP128"             "FP129"
## [130] "FP130"             "FP131"             "FP132"
## [133] "FP133"             "FP134"             "FP135"
## [136] "FP136"             "FP137"             "FP138"
## [139] "FP139"             "FP140"             "FP141"
## [142] "FP142"             "FP143"             "FP144"
## [145] "FP145"             "FP146"             "FP147"
## [148] "FP148"             "FP149"             "FP150"
## [151] "FP151"             "FP152"             "FP153"
## [154] "FP154"             "FP155"             "FP156"
## [157] "FP157"             "FP158"             "FP159"
## [160] "FP160"             "FP161"             "FP162"
## [163] "FP163"             "FP164"             "FP165"
## [166] "FP166"             "FP167"             "FP168"
## [169] "FP169"             "FP170"             "FP171"
## [172] "FP172"             "FP173"             "FP174"
## [175] "FP175"             "FP176"             "FP177"
## [178] "FP178"             "FP179"             "FP180"
## [181] "FP181"             "FP182"             "FP183"
## [184] "FP184"             "FP185"             "FP186"
## [187] "FP187"             "FP188"             "FP189"
## [190] "FP190"             "FP191"             "FP192"
## [193] "FP193"             "FP194"             "FP195"
## [196] "FP196"             "FP197"             "FP198"
## [199] "FP199"             "FP200"             "FP201"
## [202] "FP202"             "FP203"             "FP204"
## [205] "FP205"             "FP206"             "FP207"
## [208] "FP208"             "MolWeight"         "NumAtoms"
## [211] "NumNonHAtoms"      "NumBonds"          "NumNonHBonds"
## [214] "NumMultBonds"      "NumRotBonds"       "NumDblBonds"
## [217] "NumAromaticBonds"  "NumHydrogen"       "NumCarbon"
## [220] "NumNitrogen"       "NumOxygen"         "NumSulfer"
## [223] "NumChlorine"       "NumHalogen"        "NumRings"
## [226] "HydrophilicFactor" "SurfaceArea1"      "SurfaceArea2"
```

```
head(solTrainX,2)
```

```
##     FP001 FP002 FP003 FP004 FP005 FP006 FP007 FP008 FP009 FP010 FP011
## 661     0     1     0     0     1     0     0     1     0     0     0
## 662     0     1     0     1     1     1     1     1     0     0     1
##     FP012 FP013 FP014 FP015 FP016 FP017 FP018 FP019 FP020 FP021 FP022
## 661     0     0     0     1     0     0     0     1     0     0     0
## 662     0     0     0     1     1     0     1     0     0     0     0
##     FP023 FP024 FP025 FP026 FP027 FP028 FP029 FP030 FP031 FP032 FP033
## 661     0     1     0     1     0     0     0     0     0     0     0
## 662     0     0     0     0     0     1     0     0     0     0     0
##     FP034 FP035 FP036 FP037 FP038 FP039 FP040 FP041 FP042 FP043 FP044
## 661     0     0     0     0     0     1     1     0     0     0     0
## 662     0     0     0     0     0     0     0     0     0     1     0
##     FP045 FP046 FP047 FP048 FP049 FP050 FP051 FP052 FP053 FP054 FP055
## 661     0     0     0     0     0     0     0     0     0     0     0
## 662     0     1     1     0     0     0     1     0     0     0     0
##     FP056 FP057 FP058 FP059 FP060 FP061 FP062 FP063 FP064 FP065 FP066
## 661     1     0     0     0     0     0     0     1     0     1     1
## 662     0     0     0     0     1     0     0     1     1     1     0
##     FP067 FP068 FP069 FP070 FP071 FP072 FP073 FP074 FP075 FP076 FP077
## 661     1     0     1     1     0     0     0     0     0     1     0
## 662     1     1     0     1     0     1     1     1     1     1     1
##     FP078 FP079 FP080 FP081 FP082 FP083 FP084 FP085 FP086 FP087 FP088
## 661     0     1     0     0     1     0     1     0     0     1     0
## 662     1     1     1     0     1     0     1     1     0     1     1
##     FP089 FP090 FP091 FP092 FP093 FP094 FP095 FP096 FP097 FP098 FP099
## 661     1     0     1     0     0     0     0     0     1     0     0
## 662     1     1     1     0     1     0     0     0     1     0     0
##     FP100 FP101 FP102 FP103 FP104 FP105 FP106 FP107 FP108 FP109 FP110
## 661     0     1     0     0     1     0     0     0     0     0     0
## 662     0     1     1     0     1     1     1     0     0     1     0
##     FP111 FP112 FP113 FP114 FP115 FP116 FP117 FP118 FP119 FP120 FP121
## 661     0     0     0     0     0     1     0     0     0     0     0
## 662     0     0     1     0     0     1     0     1     0     0     1
##     FP122 FP123 FP124 FP125 FP126 FP127 FP128 FP129 FP130 FP131 FP132
## 661     0     0     0     0     0     0     0     0     0     0     0
## 662     0     0     0     0     1     0     0     0     0     1     0
##     FP133 FP134 FP135 FP136 FP137 FP138 FP139 FP140 FP141 FP142 FP143
## 661     0     0     0     1     0     0     0     0     0     0     0
## 662     0     1     1     0     1     1     0     0     1     1     0
##     FP144 FP145 FP146 FP147 FP148 FP149 FP150 FP151 FP152 FP153 FP154
## 661     0     0     0     0     0     0     1     0     0     0     0
## 662     0     0     0     0     0     0     0     0     0     1     1
##     FP155 FP156 FP157 FP158 FP159 FP160 FP161 FP162 FP163 FP164 FP165
## 661     0     0     0     0     0     0     0     1     0     1     0
## 662     0     0     0     0     1     0     0     1     0     1     0
##     FP166 FP167 FP168 FP169 FP170 FP171 FP172 FP173 FP174 FP175 FP176
## 661     0     0     1     0     0     0     0     0     0     0     1
## 662     1     0     1     1     1     0     0     1     0     0     0
##     FP177 FP178 FP179 FP180 FP181 FP182 FP183 FP184 FP185 FP186 FP187
## 661     0     0     1     0     0     0     0     0     0     0     0
## 662     0     1     0     0     0     0     0     0     1     0     0
##     FP188 FP189 FP190 FP191 FP192 FP193 FP194 FP195 FP196 FP197 FP198
```

```
## 661     0     0     0     0     0     0     0     0     0     0     0
## 662     0     1     0     0     0     0     0     0     0     0     0
##     FP199 FP200 FP201 FP202 FP203 FP204 FP205 FP206 FP207 FP208 MolWeight
## 661     0     0     1     0     0     0     0     0     0     0    208.28
## 662     0     0     0     1     0     0     0     0     0     0    365.54
##     NumAtoms NumNonHAtoms NumBonds NumNonHBonds NumMultBonds NumRotBonds
## 661       28           16       30           18           16           0
## 662       49           26       52           29           13           4
##     NumDblBonds NumAromaticBonds NumHydrogen NumCarbon NumNitrogen
## 661           0               16          12        14           2
## 662           0               12          23        21           3
##     NumOxygen NumSulfer NumChlorine NumHalogen NumRings HydrophilicFactor
## 661         0         0           0          0        3            -0.856
## 662         1         1           0          0        4            -0.370
##     SurfaceArea1 SurfaceArea2
## 661        25.78        25.78
## 662        52.19        80.43
```

```
# Fichero solTrainY
head(solTrainY)
```

```
## [1] -3.97 -3.98 -3.99 -4.00 -4.06 -4.08
```

2 Cuestiones sobre el fichero "EntrenamientoXY"

2.1 Construye el fichero "EntrenamientoXY" de la siguiente forma:

(a) Crea un fichero "EntrenamientoX" eliminando de "solTraintransX" todas las variables "FP..." que ocupan las 208 primeras posiciones.

```
EntrenamientoX<-data.frame(solTrainXtrans[209:ncol(solTrainXtrans)])
```

(b) Construye el fichero "EntrenamientoXY" añadiendo a "EntrenamientoX" la variable "solTrainY". (Las variables de "EntrenamientoX" serán las variables regresoras y la variable "solTrainY" la variable respuesta).

```
EntrenamientoXY<-data.frame(EntrenamientoX, solTrainY)
str(EntrenamientoXY)
```

```
## 'data.frame':    951 obs. of  21 variables:
##  $ MolWeight        : num  5.34 5.9 5.33 4.92 5.44 ...
##  $ NumAtoms         : num  3.37 3.91 3.53 3.3 3.47 ...
##  $ NumNonHAtoms     : num  2.83 3.3 2.77 2.4 2.77 ...
##  $ NumBonds         : num  3.43 3.97 3.53 3.3 3.47 ...
##  $ NumNonHBonds     : num  4.01 4.87 3.71 3.08 3.71 ...
##  $ NumMultBonds     : num  5.26 4.68 3.24 1.38 2.94 ...
##  $ NumRotBonds      : num  0 1.609 1.609 0.693 1.792 ...
##  $ NumDblBonds      : num  0 0 0.567 0.805 0 ...
##  $ NumAromaticBonds : num  2.83 2.56 1.95 0 1.95 ...
##  $ NumHydrogen      : num  3.86 5.32 4.73 4.47 4.47 ...
##  $ NumCarbon        : num  4.18 5.09 4.02 3.51 3.32 ...
##  $ NumNitrogen      : num  0.585 0.642 0 0 0.694 ...
##  $ NumOxygen        : num  0 0.693 1.099 0 0 ...
##  $ NumSulfer        : num  0 0.375 0 0 0 0.375 0 0 0 0 ...
##  $ NumChlorine      : num  0 0 0 0 0.375 ...
```

```
##  $ NumHalogen       : num  0 0 0 0 0.375 ...
##  $ NumRings         : num  1.386 1.609 0.693 0.693 0.693 ...
##  $ HydrophilicFactor: num  -1.607 -0.441 -0.385 -2.373 -0.071 ...
##  $ SurfaceArea1     : num  6.81 9.75 8.25 0 9.91 ...
##  $ SurfaceArea2     : num  6.81 12.03 8.25 0 9.91 ...
##  $ solTrainY        : num  -3.97 -3.98 -3.99 -4 -4.06 -4.08 -4.08 -4.1 -4.1 -4.11 ...
```

(c) Muestra los 5 primeros casos del fichero "EntrenamientoXY"

```
head(EntrenamientoXY, 5)
```

```
##     MolWeight NumAtoms NumNonHAtoms NumBonds NumNonHBonds NumMultBonds
## 661  5.343673 3.367296     2.833213 3.433987     4.009916     5.264609
## 662  5.904108 3.912023     3.295837 3.970292     4.871752     4.684412
## 663  5.334215 3.526361     2.772589 3.526361     3.705506     3.243492
## 665  4.921877 3.295837     2.397895 3.295837     3.076971     1.379614
## 668  5.441335 3.465736     2.772589 3.465736     3.705506     2.944766
##     NumRotBonds NumDblBonds NumAromaticBonds NumHydrogen NumCarbon
## 661   0.0000000   0.0000000         2.833213    3.862179  4.177811
## 662   1.6094379   0.0000000         2.564949    5.315193  5.092358
## 663   1.6094379   0.5670767         1.945910    4.729818  4.023944
## 665   0.6931472   0.8045302         0.000000    4.465209  3.510455
## 668   1.7917595   0.0000000         1.945910    4.465209  3.317541
##     NumNitrogen NumOxygen NumSulfer NumChlorine NumHalogen  NumRings
## 661   0.5848146 0.0000000     0.000       0.000      0.000 1.3862944
## 662   0.6423550 0.6931472     0.375       0.000      0.000 1.6094379
## 663   0.0000000 1.0986123     0.000       0.000      0.000 0.6931472
## 665   0.0000000 0.0000000     0.000       0.000      0.000 0.6931472
## 668   0.6943345 0.0000000     0.000       0.375      0.375 0.6931472
##     HydrophilicFactor SurfaceArea1 SurfaceArea2 solTrainY
## 661       -1.60654181     6.812456     6.812456     -3.97
## 662       -0.44133043     9.753834    12.029604     -3.98
## 663       -0.38485910     8.245324     8.245324     -3.99
## 665       -2.37347220     0.000000     0.000000     -4.00
## 668       -0.07098726     9.913535     9.913535     -4.06
```

2.2. Determina el modelo con una y dos variables regresoras que mejor explica la variable respuesta. Denominaremos a estos modelos M1 y M2, respectivamente.

Utilizo el comando regsubsets para realizar la selección de modelos.

Elijo los siguientes parámetros:

nvmax=2, porque el máximo tamaño del subconjunto a examinar es 2, ya que se pide el mejor modelo con 1 o 2 variables.

nbest=1, porque quiero quedarme con el mejor modelo de cada tamaño.

```
seleccion=regsubsets(solTrainY~., data=EntrenamientoXY, nvmax=2, nbest=1)
summary(seleccion)
```

```
## Subset selection object
## Call: regsubsets.formula(solTrainY ~ ., data = EntrenamientoXY, nvmax = 2,
##     nbest = 1)
## 20 Variables  (and intercept)
##                   Forced in Forced out
```

```
## MolWeight             FALSE      FALSE
## NumAtoms              FALSE      FALSE
## NumNonHAtoms          FALSE      FALSE
## NumBonds              FALSE      FALSE
## NumNonHBonds          FALSE      FALSE
## NumMultBonds          FALSE      FALSE
## NumRotBonds           FALSE      FALSE
## NumDblBonds           FALSE      FALSE
## NumAromaticBonds      FALSE      FALSE
## NumHydrogen           FALSE      FALSE
## NumCarbon             FALSE      FALSE
## NumNitrogen           FALSE      FALSE
## NumOxygen             FALSE      FALSE
## NumSulfer             FALSE      FALSE
## NumChlorine           FALSE      FALSE
## NumHalogen            FALSE      FALSE
## NumRings              FALSE      FALSE
## HydrophilicFactor     FALSE      FALSE
## SurfaceArea1          FALSE      FALSE
## SurfaceArea2          FALSE      FALSE
## 1 subsets of each size up to 2
## Selection Algorithm: exhaustive
##          MolWeight NumAtoms NumNonHAtoms NumBonds NumNonHBonds
## 1  ( 1 ) "*"       " "      " "           " "      " "
## 2  ( 1 ) " "       " "      "*"           " "      " "
##          NumMultBonds NumRotBonds NumDblBonds NumAromaticBonds NumHydrogen
## 1  ( 1 ) " "          " "         " "         " "               " "
## 2  ( 1 ) " "          " "         " "         " "               " "
##          NumCarbon NumNitrogen NumOxygen NumSulfer NumChlorine NumHalogen
## 1  ( 1 ) " "       " "         " "       " "       " "         " "
## 2  ( 1 ) " "       " "         " "       " "       " "         " "
##          NumRings HydrophilicFactor SurfaceArea1 SurfaceArea2
## 1  ( 1 ) " "      " "               " "          " "
## 2  ( 1 ) " "      " "               "*"          " "
```

Tras examinar los resultados comprobamos que el mejor modelo con una variable es aquel que tiene como variable regresora **MolWeight**. Por tanto utilizaré la función lm para construir el modelo M1

```
M1=lm(solTrainY~MolWeight, data=EntrenamientoXY)
summary(M1)
```

```
##
## Call:
## lm(formula = solTrainY ~ MolWeight, data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0374 -0.7107  0.2403  0.9619  5.5361
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.9542     0.5466   21.87   <2e-16 ***
## MolWeight    -2.8222     0.1047  -26.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.541 on 949 degrees of freedom
## Multiple R-squared:  0.4337, Adjusted R-squared:  0.4331
## F-statistic: 726.7 on 1 and 949 DF,  p-value: < 2.2e-16
```

Del mismo modo, observando los resultados obtenido con el comando regsubset, vemos que el mejor modelo con dos variables es el que tiene como variables explicativas **NumNonHAtoms** y **SurfaceArea1**

```
M2=lm(solTrainY~NumNonHAtoms+SurfaceArea1, data=EntrenamientoXY)
summary(M2)
```

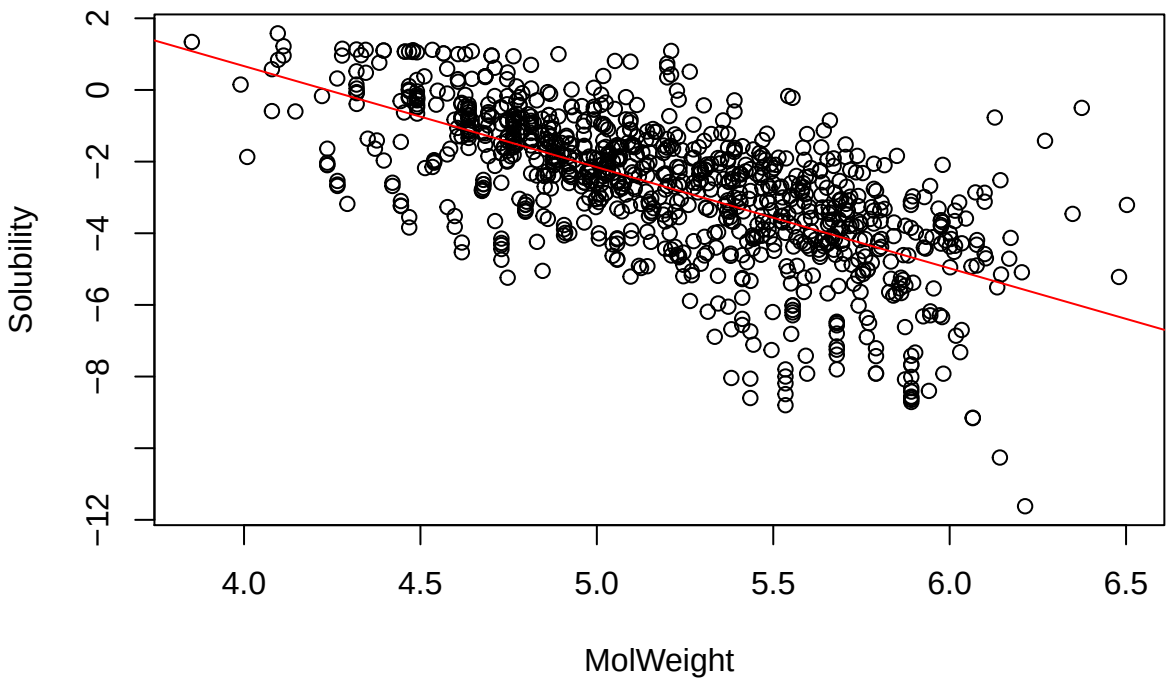
```
##
## Call:
## lm(formula = solTrainY ~ NumNonHAtoms + SurfaceArea1, data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2475 -0.6472 -0.0138  0.6114  4.0462
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.618947   0.189896   29.59   <2e-16 ***
## NumNonHAtoms -4.143351   0.080834  -51.26   <2e-16 ***
## SurfaceArea1  0.331360   0.008162   40.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.005 on 948 degrees of freedom
## Multiple R-squared:  0.7592, Adjusted R-squared:  0.7587
## F-statistic:  1495 on 2 and 948 DF,  p-value: < 2.2e-16
```

2.3 En los modelos M1 y M2

(a) Representa los gráficos de dispersión de los regresores frente a la respuesta.

Modelo M1

```
plot(solTrainY~MolWeight, data = EntrenamientoXY, xlab = "MolWeight",
     ylab = "Solubility", main="2.3.a (modelo M1): Gráfica de Dispersión MolWeight")
abline(M1, col = "red")
```

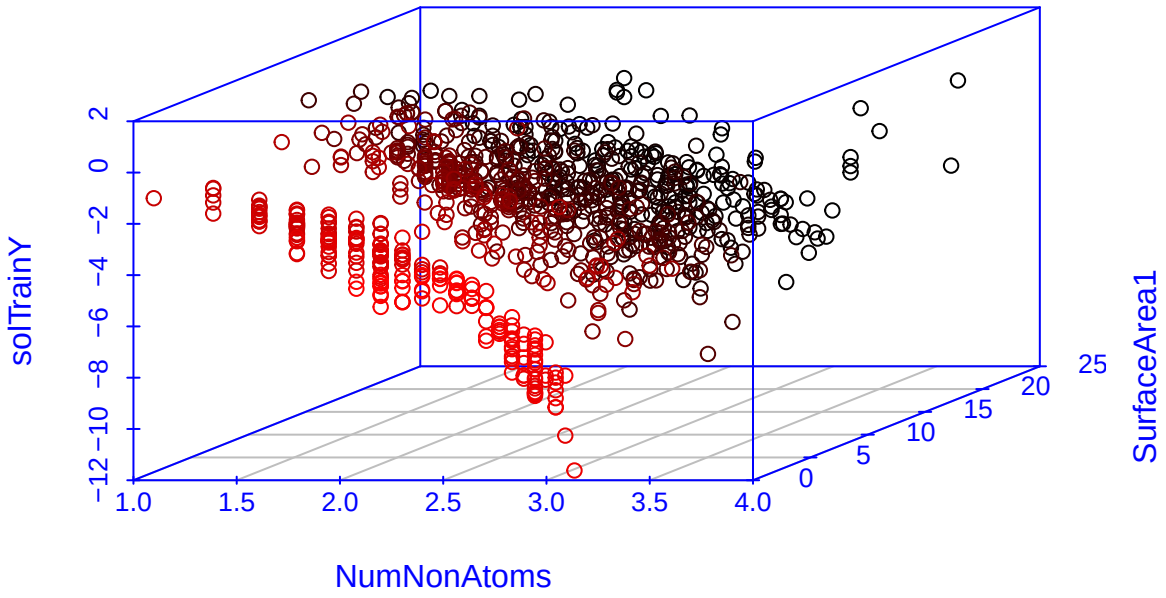


Modelo M2

Este modelo tiene dos variables regresoras, en primer lugar mostraremos la gráfica 3D con ambas variables regresoras, y luego se construirá las gráficas de dispersion de cada una de las variables explicativas frente a la variable respuesta.

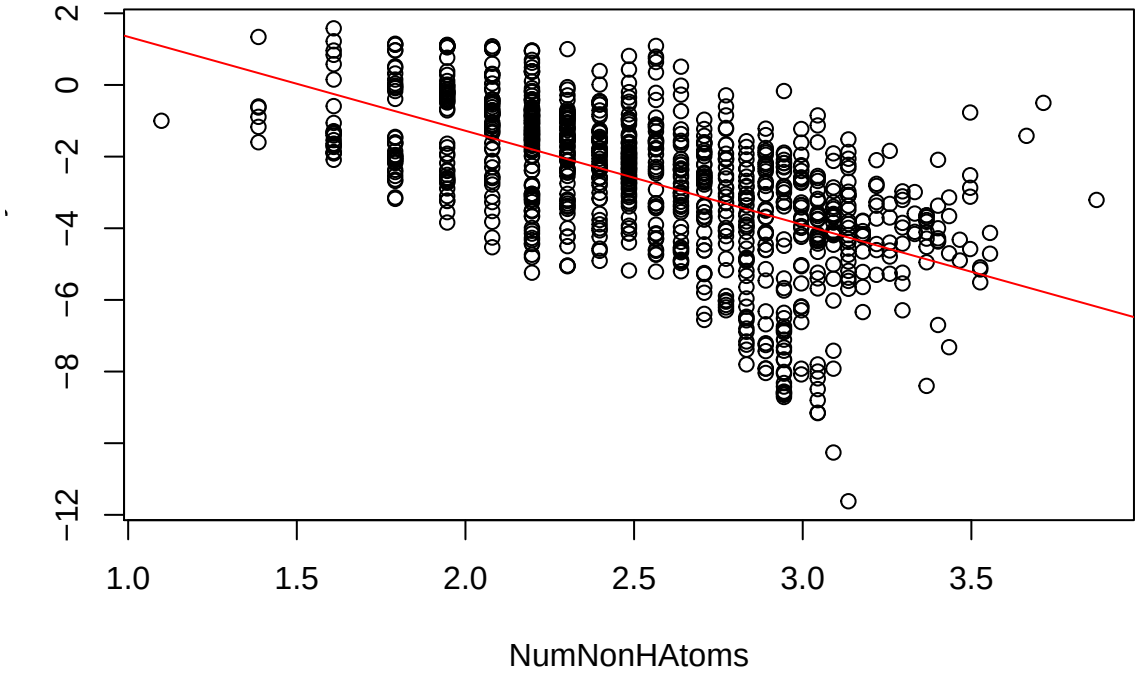
```
scatterplot3d(EntrenamientoXY$NumNonHAtoms, EntrenamientoXY$SurfaceArea1, EntrenamientoXY$solTrainY,
              xlab="NumNonAtoms", ylab="SurfaceArea1", zlab="solTrainY",
              highlight.3d=TRUE, col.axis = "blue", col.lab = "blue",
              main="2.3.a (modelo M2): Gráfica dispersión 3D")
```

2.3.a (modelo M2): Gráfica dispersión 3D



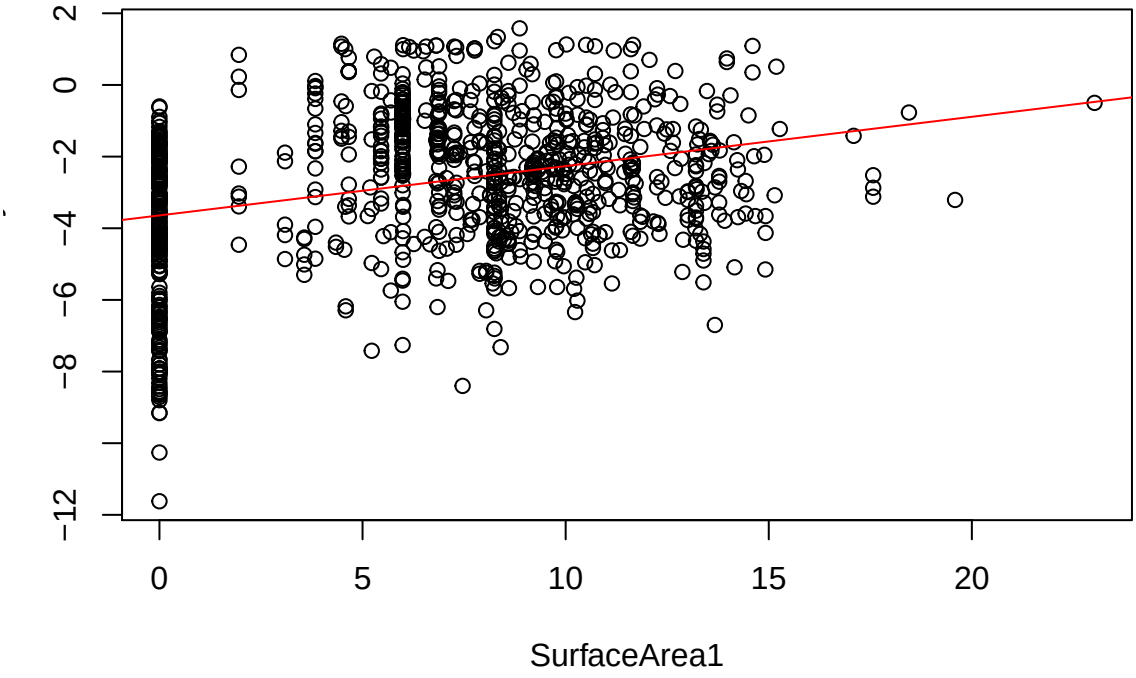

```
plot(solTrainY~NumNonHAtoms, data = EntrenamientoXY, xlab = "NumNonHAtoms",
     ylab = "Solubility", main="1.3.a (modelo M2): Gráfica de Dispersión NumNonHAtoms")
M2_1=lm(solTrainY~NumNonHAtoms, data = EntrenamientoXY)
abline(M2_1, col = "red")
```

1.3.a (modelo M2): Gráfica de Dispersión NumNonHAtoms




```
plot(solTrainY~SurfaceArea1, data = EntrenamientoXY, xlab = "SurfaceArea1",
     ylab = "Solubility", main="2.3.a (modelo M2): Gráfica de Dispersión SurfaceArea1")
M2_2=lm(solTrainY~SurfaceArea1, data = EntrenamientoXY)
abline(M2_2, col = "red")
```

2.3.a (modelo M2): Gráfica de Dispersión SurfaceArea1



(b) Obtén los EMC de los parámetros, los intervalos de confianza para los parámetros y los p-valores asociados a los tests.

Modelo M1

```
# EMC de los parámetros y p-valores
summary(M1)
```

```
##
## Call:
## lm(formula = solTrainY ~ MolWeight, data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0374 -0.7107  0.2403  0.9619  5.5361
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.9542     0.5466   21.87   <2e-16 ***
## MolWeight    -2.8222     0.1047  -26.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.541 on 949 degrees of freedom
## Multiple R-squared:  0.4337, Adjusted R-squared:  0.4331
## F-statistic: 726.7 on 1 and 949 DF,  p-value: < 2.2e-16
```

```r
# Intervalos de confianza
confint(M1, level=0.95)
```

```
##                 2.5 %    97.5 %
## (Intercept) 10.881503 13.026849
## MolWeight   -3.027673 -2.616757
```

Modelo M2

```r
# EMC de los parámetros y p-valores
summary(M2)
```

```
##
## Call:
## lm(formula = solTrainY ~ NumNonHAtoms + SurfaceArea1, data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2475 -0.6472 -0.0138  0.6114  4.0462
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.618947   0.189896   29.59   <2e-16 ***
## NumNonHAtoms -4.143351   0.080834  -51.26   <2e-16 ***
## SurfaceArea1  0.331360   0.008162   40.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.005 on 948 degrees of freedom
## Multiple R-squared:  0.7592, Adjusted R-squared:  0.7587
## F-statistic:  1495 on 2 and 948 DF,  p-value: < 2.2e-16
```

```r
# Intervalos de confianza
confint(M2, level=0.95)
```

```
##                   2.5 %     97.5 %
## (Intercept)   5.2462814  5.9916119
## NumNonHAtoms -4.3019857 -3.9847166
## SurfaceArea1  0.3153423  0.3473768
```

(c) Interpreta el significado de los EMC

Modelo M1 (lineal simple)

De forma genérica: E(Yx) = Beta0 + Beta1 * x

Beta0 = E (Yx=0)

Beta1 = E (Yx+1) - E (Yx)

Por tanto:

E(solTrainY) = Beta0 + Beta1 * MolWeight = 11.9542 - 2.8222 * MolWeight

Beta0 = 11.9542. Es el valor esperado de la solubilidad cuando la variable MolWeight es igual a 0.

Beta1 = - 2.8222. Es el incremento de la solubilidad cuando MolWeight aumenta en una unidad. Es decir, si MolWeight aumenta una unidad la solubilidad decrece 2.822215 unidades.

Modelo M2 (lineal múltiple)

E(solTrainY) = Beta0 + Beta1 * NumNonHAtoms + Beta2 * SurfaceArea1 =

```
5.6189467 - 4.1433512 * NumNonHAtoms + 0.3313595 * SurfaceArea1
```

Beta0 = 5.6189467, es el valor esperado de solTrainY cuando NumNonHAtoms y SurfaceArea1 toma el valor 0.

Beta1 = - 4.1433512, muestra la relación entre solTrainY y NumNonHAtoms. El incremento en una unidad en NumNonHAtoms provoca un decremento de 4.1433512 unidades en solTrainY.

Beta2 = 0.3313595, por tanto un incremento en una unidad en SurfaceArea1 provoca un aumento en 0.3313595 unidades en solTrainY

(d) ¿Qué conclusiones se obtienen de los p-valores resultantes?

Modelo M1

El p-valor representa la probabilidad de que se cumpla la hipótesis nula, es decir, que la variable MolWeight no influya sobre la variable solTrainY. La probabilidad obtenida es mucho menor que 0.005. Por tanto, podemos rechazar la hipótesis nula (Beta1=0) y concluir que la variable MolWeight es significativa.

Modelo M2

Se obtiene una probabilidad << 0.005, por tanto podemos rechazar la hipotesis nula, que representa que las variables NumNonHAtoms y SurfaceArea1 no influyen sobre la variable objetivo solTrainY. Podemos concluir que ambas son significativas porque no tenemos evidencia de que sean nulas.

2.4 En el modelo M2, determina la estimación del valor medio de la respuesta, el intervalo de confianza y el intervalo de predicción (ambos al 95%) correspondiente a los valores (3, 7) de los regresores. (Justifica que (3, 7) es un valor apropiado para realizar lo que se pide). Presentar los datos resultantes en una tabla del tipo (con sólo dos decimales)

En primer lugar comprobamos si el valor (3,7) pertenece al rango y por tanto es apropiado

```
# El valor 3 está en el rango
range_numnonhatoms=range(EntrenamientoXY$NumNonHAtoms)
range_numnonhatoms
```

```
## [1] 1.098612 3.871201
```

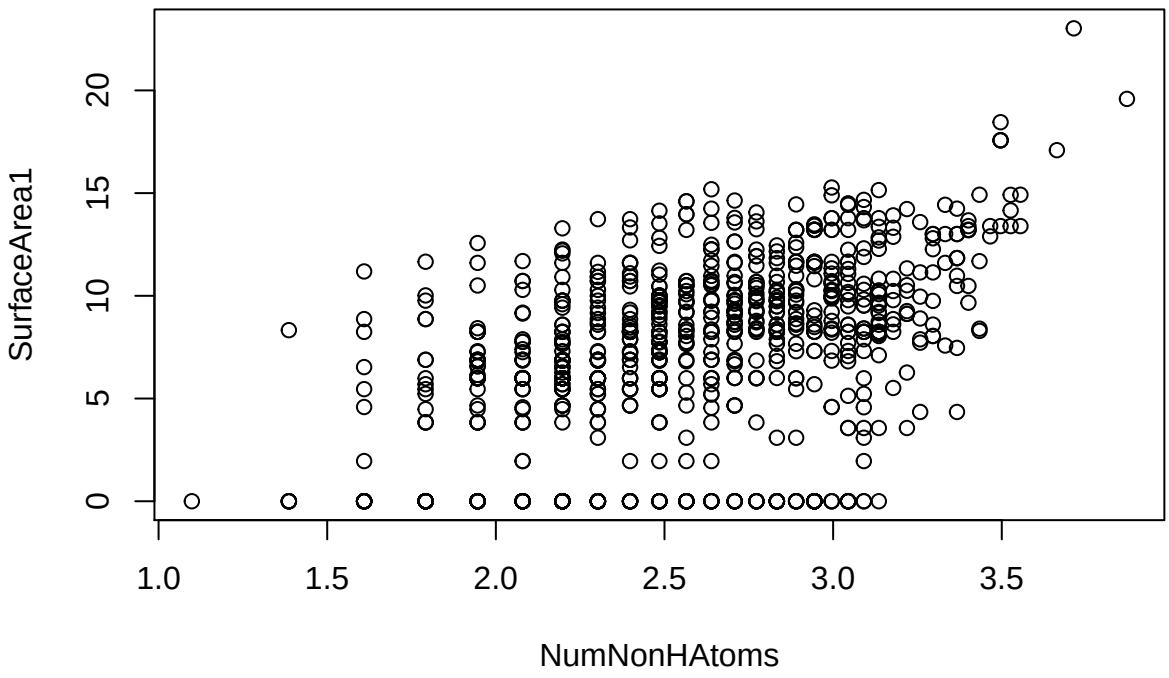
```
# El valor 7 está en el rango
range_surfacearea1=range(EntrenamientoXY$SurfaceArea1)
range_surfacearea1
```

```
## [1]  0.00000 23.02034
```

```
range_entrenamientoXY=range(EntrenamientoXY)
range_entrenamientoXY
```

```
## [1] -11.62000  23.02034
```

```
plot(EntrenamientoXY$NumNonHAtoms, EntrenamientoXY$SurfaceArea1,
     xlab="NumNonHAtoms", ylab="SurfaceArea1")
```



Se observa que ambos valores (3, 7) pertenecen al rango calculado, y por tanto se continua con los siguiente cálculos.

```
# Valor medio de la respuesta
# E(solTrainY) = 5.6189467 - 4.1433512 * NumNonHAtoms + 0.3313595* SurfaceArea1
E1 =  5.6189467 - 4.1433512 * 3 + 0.3313595 * 7
E1
```

```
## [1] -4.49159
```

```
# Intervalo de confianza
IC=as.data.frame(predict(M2, data.frame(NumNonHAtoms=3, SurfaceArea1=7),
                         interval="confidence", level=0.95))
IC
```

```
##        fit       lwr       upr
## 1 -4.49159 -4.586087 -4.397094
```

```
# Intervalo de predicción
IP=as.data.frame(predict(M2, data.frame(NumNonHAtoms=3, SurfaceArea1=7),
                         interval="prediction", level=0.95))
IP
```

```
##        fit       lwr       upr
## 1 -4.49159 -6.466732 -2.516448
```

El intervalo de confianza informa del valor medio de Y para una X dada. Mientras que el Intervalo de predicción informa del rango de valores de Y para un X en particular

Existen 2 diferencias entre el intervalo de predicción y de confianza

1) El intervalo de predicción utiliza la desviación estándar en lugar del error estándar del intervalo de confianza. Como la desviación típica es siempre mayor que el error estándar, los intervalos predictivos serán siempre más amplios que los de confianza para el mismo nivel de incertidumbre.

2) Para calcular el intervalo de confianza tenemos que medir previamente el valor en una o varias muestras, mientras que el intervalo predictivo se calcula a priori, antes de extraer el sujeto o sujetos de la población.

A continuación presentamos los resultados en el formato de tabla solicitado

```
x <- data.frame(row.names=c("(3,7)"))
# Valor de Estimacion
x[,1] <- E1
# Intervalo de confianza
x[,2] <- paste(round(IC$lwr, 2) , ",", round(IC$upr, 2), sep = "")
# Intervalo de predicción
x[,3] <- paste(round(IP$lwr, 2) , ",", round(IP$upr, 2), sep = "")

print(knitr::kable(x, format = "pandoc",
                   col.names = c("Estimación", "IC(95%)", "IP(95%)"), align='c'))
```

```
##
##
##          Estimación     IC(95%)        IP(95%)
## ------  ------------  ------------  -------------
## (3,7)     -4.49159     -4.59,-4.4    -6.47,-2.52
```

2.5 Modelo con interacción

(a) Construye el modelo M2int, resultante de añadir a M2 la interacción entre sus dos regresores.

```
M2int=lm(solTrainY~NumNonHAtoms*SurfaceArea1, data=EntrenamientoXY)
summary(M2int)
```

```
##
## Call:
## lm(formula = solTrainY ~ NumNonHAtoms * SurfaceArea1, data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1003 -0.6468  0.0018  0.6054  4.0761
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 6.17875    0.29725  20.786  < 2e-16 ***
## NumNonHAtoms               -4.36885    0.12254 -35.652  < 2e-16 ***
## SurfaceArea1                0.23981    0.03834   6.255 6.01e-10 ***
## NumNonHAtoms:SurfaceArea1   0.03486    0.01426   2.443   0.0147 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.003 on 947 degrees of freedom
## Multiple R-squared:  0.7607, Adjusted R-squared:   0.76
## F-statistic:  1004 on 3 and 947 DF,  p-value: < 2.2e-16
```

(b) Construye la siguiente tabla comparativa: M1, M2, M2int

```
M1_info=summary(M1)
M2_info=summary(M2)
M2int_info=summary(M2int)

v_m1=c(M1_info$r.squared, M1_info$adj.r.squared, M1_info$sigma)
v_m2=c(M2_info$r.squared, M2_info$adj.r.squared, M2_info$sigma)
v_m2int=c(M2int_info$r.squared, M2int_info$adj.r.squared, M2int_info$sigma)

comp <- data.frame(row.names=c("R2", "R2 ajustado", "Residual standard error"))
# M1
comp[,1] <- v_m1
# M2
comp[,2] <- v_m2
# M3
comp[,3] <- v_m2int


print(knitr::kable(round(comp, 3), format = "pandoc",
                   col.names = c("M1", "M2", "M2int"), align='c'))
```

```
##
##
##                            M1       M2      M2int
## ------------------------  -------  -------  -------
## R2                         0.434    0.759    0.761
## R2 ajustado                0.433    0.759    0.760
## Residual standard error    1.541    1.005    1.003
```

Los valores de R2 y RSE obtenidos para el modelo M1 no son buenos. La inclusion de una variable más en el modelo M2 mejora significativamente los valores de R2 y disminuye el error. Sin embargo, la inclusión de una nueva variable en el modelo M2int mejora ligeramente, pero dicha mejora no es suficiente como para justificar la inclusion de una variable más.

En conclusión, el modelo con el que mejores resultados se obtienen, y por tanto mejor se ajusta a los datos es el modelo M2.

2.6 Construye el modelo de regresión lineal con todos los regresores disponibles en el fichero (Mtodas).

(a) Interpretar los resultados obtenidos en comparación con los de los modelos M1, M2 y M2int considerados en los apartados anteriores.

Construimos el modelo que incluye todas las varibles

```
Mtodas=lm(solTrainY~., data=EntrenamientoXY)
(Mtodas_info=summary(Mtodas))
```

```
##
## Call:
## lm(formula = solTrainY ~ ., data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2090 -0.4747  0.0308  0.5025  3.2191
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        9.15400    1.78178   5.138 3.39e-07 ***
## MolWeight         -1.70828    0.25078  -6.812 1.73e-11 ***
## NumAtoms          -5.63193    2.58328  -2.180 0.029496 *
## NumNonHAtoms      10.15939    2.68088   3.790 0.000161 ***
## NumBonds          -0.21465    2.01085  -0.107 0.915012
## NumNonHBonds      -5.21651    1.48013  -3.524 0.000445 ***
## NumMultBonds      -0.52974    0.09340  -5.672 1.89e-08 ***
## NumRotBonds       -0.17659    0.08873  -1.990 0.046864 *
## NumDblBonds       -0.50286    0.14929  -3.368 0.000787 ***
## NumAromaticBonds   0.47356    0.11774   4.022 6.23e-05 ***
## NumHydrogen        1.35525    0.17988   7.534 1.16e-13 ***
## NumCarbon         -0.09082    0.23838  -0.381 0.703310
## NumNitrogen        1.58317    0.20128   7.866 1.02e-14 ***
## NumOxygen          1.13675    0.14146   8.036 2.81e-15 ***
## NumSulfer         -0.73686    0.38475  -1.915 0.055781 .
## NumChlorine       -0.92794    0.32532  -2.852 0.004436 **
## NumHalogen         0.46968    0.34545   1.360 0.174276
## NumRings           1.66077    0.41083   4.042 5.73e-05 ***
## HydrophilicFactor -0.16234    0.05596  -2.901 0.003810 **
## SurfaceArea1       0.10527    0.03835   2.745 0.006172 **
## SurfaceArea2       0.06735    0.03370   1.999 0.045945 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.817 on 930 degrees of freedom
## Multiple R-squared:  0.844,  Adjusted R-squared:  0.8406
## F-statistic: 251.6 on 20 and 930 DF,  p-value: < 2.2e-16
```

Representamos los resultados obtenidos en una tabla comparativa

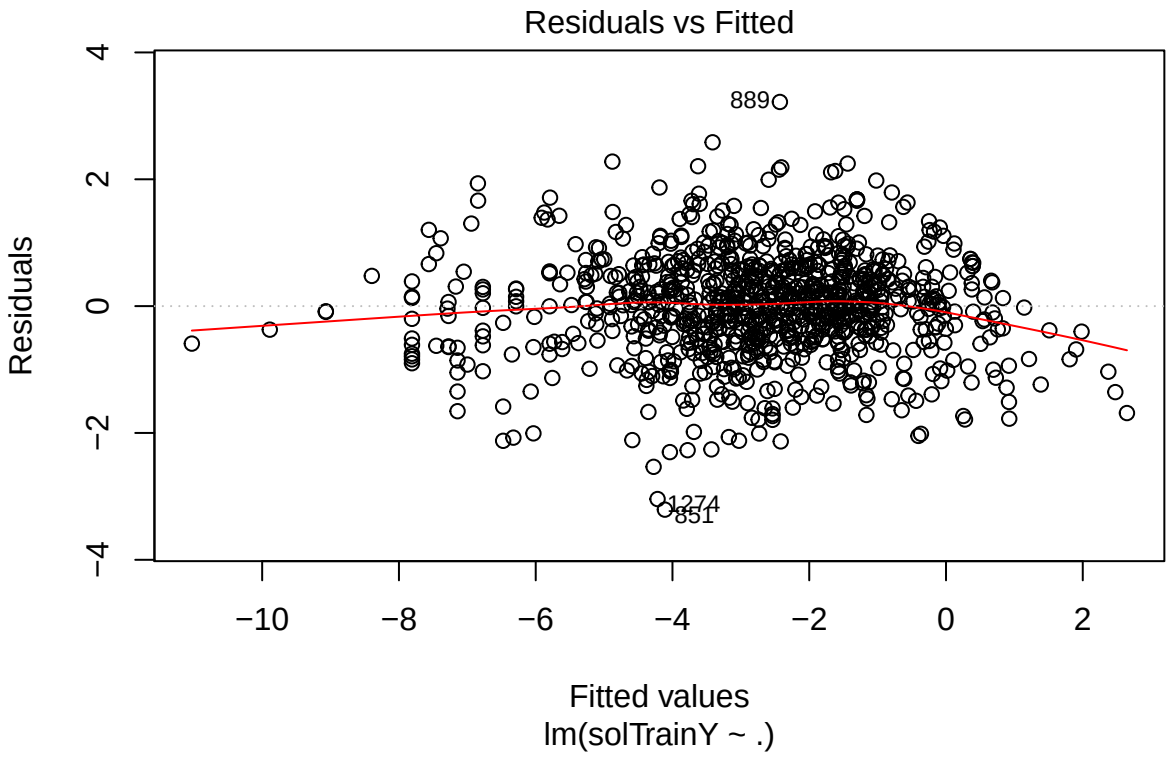
```
v_mtodas=c(Mtodas_info$r.squared, Mtodas_info$adj.r.squared, Mtodas_info$sigma)
comp[,4]<-v_mtodas
print(knitr::kable(round(comp, 3), format = "pandoc",
                   col.names = c("M1", "M2", "M2int", "Mtodas"), align='c'))
```

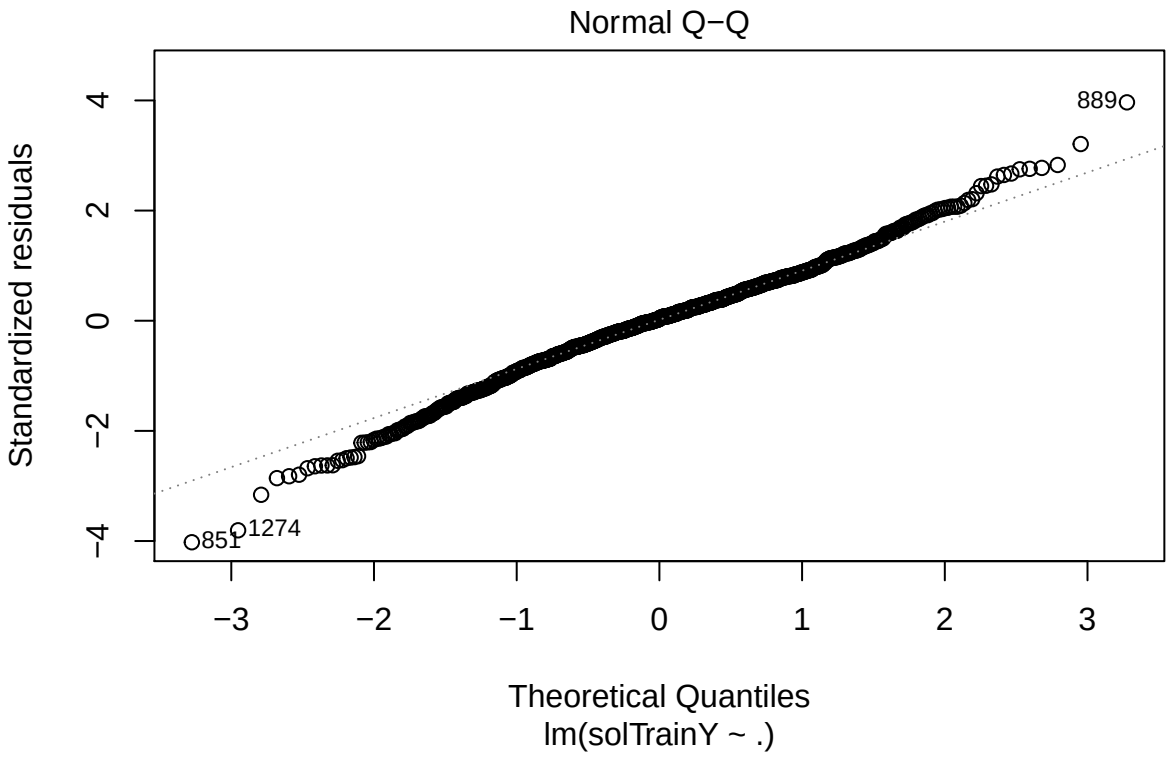
```
##
##
##                            M1       M2      M2int    Mtodas
## ------------------------  -------  -------  -------  --------
## R2                         0.434    0.759    0.761    0.844
## R2 ajustado                0.433    0.759    0.760    0.841
## Residual standard error    1.541    1.005    1.003    0.817
```

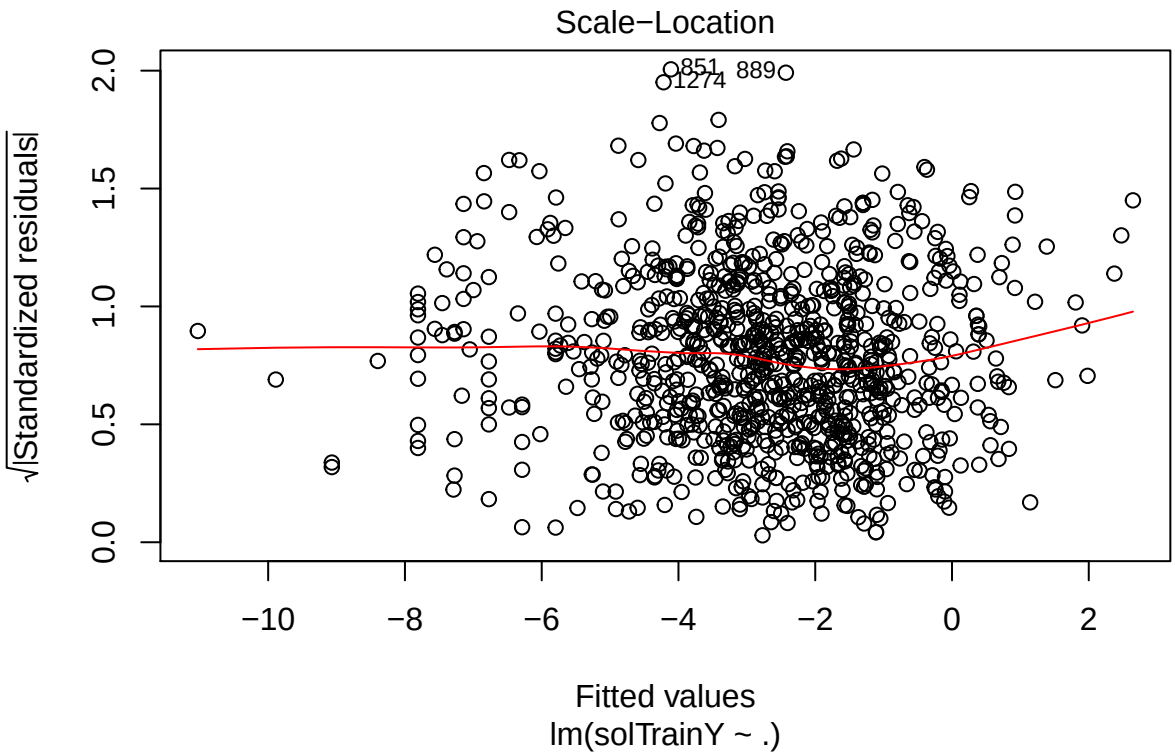
Observamos que para el modelo que incluye todas las variables regresoras se obtine un R2 mayor y un error menor que con el resto de modelos que estamos estudiando. Sin embargo al añadir todas las variables regresoras hemos añadido mucha más complejidad, puesto que con M2 teníamos un modelo de 2 variables y con Mtodas tenemos un modelo de 20 variables. Sería interesante continuar con el estudio y analizar si es posible simplificar el modelo Mtodas seleccionando variables y eliminando aquellas que no sean significativas para nuestro estudio.

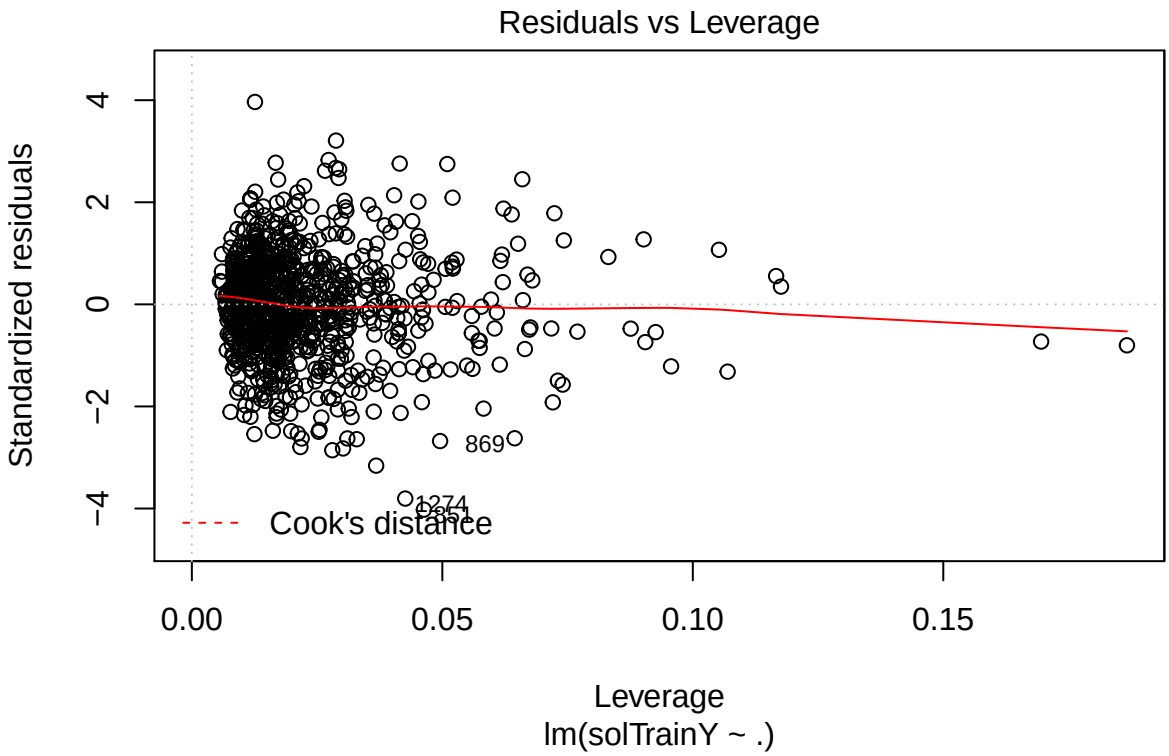
(b) Obtén los gráficos de diagnósticos y comenta los resultados

```
#par(mfrow=c(2,2))
plot(Mtodas)
```


Residuals vs Fitted
Residuals
Fitted values
lm(solTrainY ~ .)
889
1274
851
4
2
0
−2
−4
−10
−8
−6
−4
−2
0
2


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(solTrainY ~ .)
889
1274
851
−4
−2
0
2
4
−3
−2
−1
0
1
2
3


Scale−Location
851
889
1274
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
−10
−8
−6
−4
−2
0
2
Fitted values
lm(solTrainY ~ .)



Las siguientes gráficas nos ayudarán a diagnosticar y validar el modelo obtenido. Veamos la interpretación de cada una de ellas:

Residuals vs Fitted

Esta gráfica muestra si los residuos tienen patrones no lineales, es decir si existe una relacion no lineal entre las variables regresoras y la variable respuesta. Por tanto nos ayuda a verficar la hipótesis de linealidad

En este caso observamos que los puntos se distribuyen sin ningún patron distintivo alrededor de la linea roja, lo cual indica que no existen patrones no lineales.

Normal Q-Q

Esta gráfica muestra si los residuos siguen una distribución normal, por tanto nos ayuda a verficar hipótesis de normalidad.

En el modelo Mtodas los residuos siguen bien la recta, no se aprecian desviaciones significativas, salvo los siguientes 3 puntos que se marcan en la gráfica: 851, 889 y 1274. Por tanto podemos verificar que se cumple la hipótesis de normalidad.

Scale location

Esta gráfica muestra si los residuos se distribuyen por igual a lo largo de los rangos de predictores, por tanto nos sirve para verificar hipótesis de igualdad de varianza (homoscedasticidad).

En el caso que nos ocupa vemos que los puntos se distribuyen igualmente alredor de la línea horizontal. La línea roja es horizontal y no tiene pendiente, lo cual verifica la hipótesis de homocedasticidad.

Residuals vs Leverage

Este gráfico nos ayuda a determinar las observaciones influencia (observaciones que afectan mucho al modelo).

En nuestro caso no hay ningún caso influyente, todos los puntos están dentro de la linea de distancia de Cook.

(c) Calcula los vif (factores de inflación de la varianza) de los regresores ¿Qué conclusión se obtiene?

```
vif(Mtodas)
```

```
##         MolWeight          NumAtoms      NumNonHAtoms          NumBonds
##         20.414139       1983.747293       2119.826887       1371.838151
##      NumNonHBonds      NumMultBonds       NumRotBonds       NumDblBonds
##       2420.464035         37.148038          5.788255          4.629760
##  NumAromaticBonds       NumHydrogen         NumCarbon       NumNitrogen
##         25.132366         64.370392         78.634635          4.327202
##         NumOxygen         NumSulfer       NumChlorine        NumHalogen
##         11.216935          3.608682          4.730450          6.419813
##          NumRings HydrophilicFactor      SurfaceArea1      SurfaceArea2
##         69.691641          4.833255         42.558406         35.277628
```

Valores VIF > 10 es indicativo de serios problemas de multicolinealidad.

Por tanto podemos concluir que las siguientes variables regresoras presentar problemas de multicolinealidad:

MolWeight, NumAtoms, NumNonHAtoms, NumBonds, NumNonHBonds, NumMultBonds, NumAromaticBonds, NumHydrogen, NumCarbon, NumOxygen, NumRings, SurfaceArea1 y SurfaceArea2

Como solución se podrían eliminar las variables regresoras problemáticas, ya que la información de estas variables no suele afectar mucho puesto que están incluidas en las otras, y por tanto es redundante.

2.7 Determina el modelo resultante de una regresión paso a paso hacia adelante. Sea MHA el modelo resultante.

```
# Modelo nulo, solo tiene el termino independiente
null=lm(solTrainY~1, data=EntrenamientoXY)
# Modelo con todas las variables regresoras
full=Mtodas
# Regresión paso a paso hacia adelante
MHA=stepAIC(null, scope=list(lower=null, upper=full), direction="forward", trace=0)
(MHA_info=summary(MHA))
```

```
##
## Call:
## lm(formula = solTrainY ~ MolWeight + SurfaceArea1 + NumNonHAtoms +
##     NumHydrogen + NumAtoms + NumRings + NumCarbon + NumOxygen +
##     NumNitrogen + NumDblBonds + NumMultBonds + NumAromaticBonds +
##     NumNonHBonds + NumChlorine + HydrophilicFactor + NumRotBonds,
##     data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2804 -0.4839  0.0453  0.5090  3.2398
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        9.47699    1.70836   5.547 3.78e-08 ***
## MolWeight         -1.59701    0.19812  -8.061 2.31e-15 ***
## SurfaceArea1       0.16512    0.02551   6.472 1.56e-10 ***
## NumNonHAtoms      10.11639    2.58316   3.916 9.65e-05 ***
```

```
## NumHydrogen         1.39124    0.15078   9.227  < 2e-16 ***
## NumAtoms           -6.27533    0.72706  -8.631  < 2e-16 ***
## NumRings            1.58729    0.40026   3.966 7.88e-05 ***
## NumCarbon           0.03595    0.20041   0.179 0.857673
## NumOxygen           1.16275    0.13774   8.442  < 2e-16 ***
## NumNitrogen         1.61887    0.19784   8.183 9.04e-16 ***
## NumDblBonds        -0.54813    0.14449  -3.793 0.000158 ***
## NumMultBonds       -0.54910    0.09195  -5.972 3.34e-09 ***
## NumAromaticBonds    0.46403    0.11664   3.978 7.48e-05 ***
## NumNonHBonds       -5.16106    1.41903  -3.637 0.000291 ***
## NumChlorine        -0.62528    0.21622  -2.892 0.003919 **
## HydrophilicFactor -0.15343    0.05478  -2.801 0.005206 **
## NumRotBonds        -0.17029    0.08716  -1.954 0.051030 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.818 on 934 degrees of freedom
## Multiple R-squared:  0.8429, Adjusted R-squared:  0.8402
## F-statistic: 313.3 on 16 and 934 DF,  p-value: < 2.2e-16
```

El modelo resultante de la regresión paso a paso está compuesto por 16 variables regresoras.

El p-valor obtenido para la variable NumCarbon es igual a 0.857673> 0.05, con lo cual aceptamos la hipótesis nula, y por tanto podemos concluir que la variable NumCarbon no es significativa y puede ser eliminada del modelo.

A continuación representamos los resultados obtenidos para el modelo MHA en la tabla comparativa

```
v_mHA=c(MHA_info$r.squared, MHA_info$adj.r.squared, MHA_info$sigma)
comp[,5]<-v_mHA
print(knitr::kable(round(comp, 3), format = "pandoc",
                   col.names = c("M1", "M2", "M2int", "Mtodas", "MHA"), align='c'))
```

```
##
##
##                             M1       M2      M2int    Mtodas     MHA
## ------------------------  -------  -------  -------  --------  -------
## R2                         0.434    0.759    0.761    0.844     0.843
## R2 ajustado                0.433    0.759    0.760    0.841     0.840
## Residual standard error    1.541    1.005    1.003    0.817     0.818
```

Observamos que el valor de R2 y RSE son muy similares a los obtenidos por Mtodas, con la ventaja de que el modelo MHA tiene 16 variables regresoras y el modelo Mtodas tiene 20. Con lo cual podemos concluir que el modelo MHA es mejor que Mtodas. Continua siendo interesante experimentar con los datos para determinar si es posible disminuir la complejidad del modelo eliminando variables explicativas, como por ejemplo variable NumCarbon, que no parece significativas y otras variable con problemas de multicolinealidad.

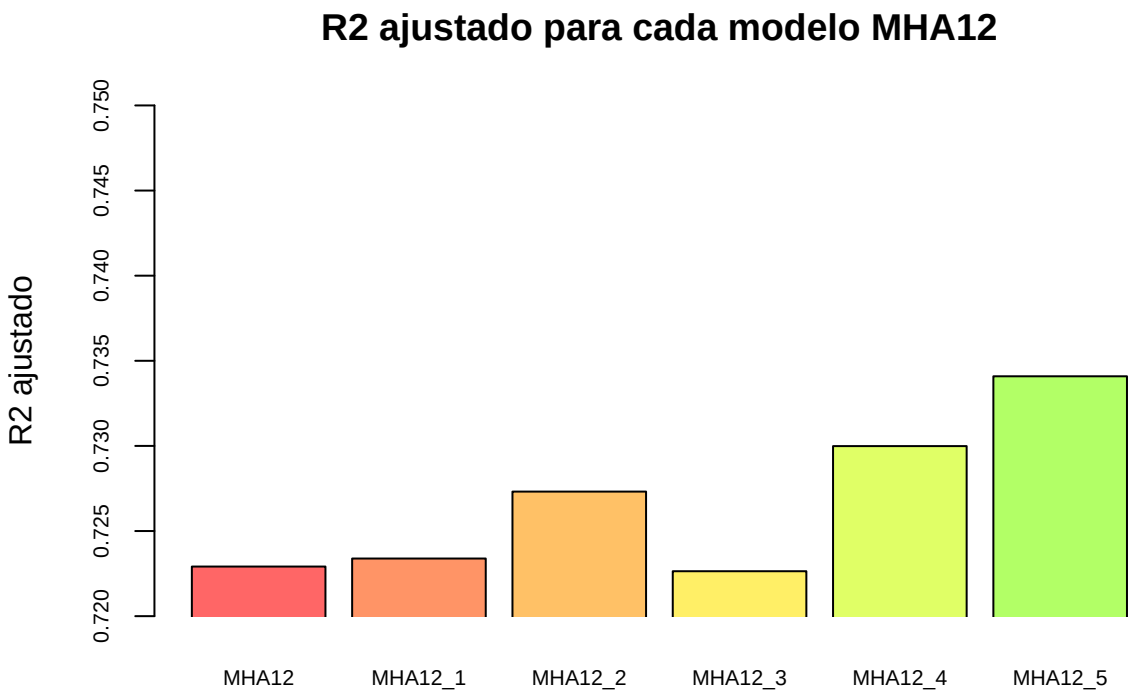
2.8. Construye un gráfico para comparar R2 ajustado en los siguientes modelos, y comenta los resultados.

MHA12: Modelo con las dos primeras variables (v1 y v2) que entran en la regresión paso a paso hacia adelante

```
# Modelo con las dos primeras variables (v1 y v2) regresión paso a paso hacia adelante
MHA12=stepAIC(null, scope=list(lower=null, upper=full), direction="forward", steps=2, trace=0)
summary(MHA12)
```

```
##
## Call:
## lm(formula = solTrainY ~ MolWeight + SurfaceArea1, data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5320 -0.5900 -0.0318  0.6024  3.7656
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  14.212508   0.388783   36.56   <2e-16 ***
## MolWeight    -3.588844   0.077126  -46.53   <2e-16 ***
## SurfaceArea1  0.257523   0.008169   31.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.077 on 948 degrees of freedom
## Multiple R-squared:  0.7235, Adjusted R-squared:  0.7229
## F-statistic:  1240 on 2 and 948 DF,  p-value: < 2.2e-16
```

```r
# MHA12+(v1)2
MHA12_1=lm(solTrainY ~ MolWeight + SurfaceArea1+ I(MolWeight^2), data = EntrenamientoXY)

# MHA12+(v1)2+(v1)3
MHA12_2=lm(solTrainY ~ MolWeight + SurfaceArea1+ I(MolWeight^2) + I(MolWeight^3),
           data = EntrenamientoXY)

# MHA12+(v2)2
MHA12_3=lm(solTrainY ~ MolWeight + SurfaceArea1+ I(SurfaceArea1^2), data = EntrenamientoXY)

# MHA12+(v2)2+(v2)3
MHA12_4=lm(solTrainY ~ MolWeight + SurfaceArea1+ I(SurfaceArea1^2) + I(SurfaceArea1^3),
           data = EntrenamientoXY)

# MHA12+(v1 × v2) + (v1)2 + (v1)3 + (v2)2 + (v2)3
MHA12_5=lm(solTrainY ~ MolWeight * SurfaceArea1 + I(MolWeight^2) + I(MolWeight^3) +
             I(SurfaceArea1^2) + I(SurfaceArea1^3), data = EntrenamientoXY)


y<-c((summary(MHA12)$adj.r.squared), (summary(MHA12_1)$adj.r.squared),
     (summary(MHA12_2)$adj.r.squared), (summary(MHA12_3)$adj.r.squared),
     (summary(MHA12_4)$adj.r.squared), (summary(MHA12_5)$adj.r.squared))

barplot(y,
        main="R2 ajustado para cada modelo MHA12",
        ylab="R2 ajustado",
        names=c("MHA12", "MHA12_1", "MHA12_2", "MHA12_3", "MHA12_4", "MHA12_5"),
        col=rainbow(20, alpha = .6),
        ylim=c(0.72, 0.75),
        xpd=F,
        cex.axis=0.7, cex.names=0.7)
```



Observamos que de todos los modelos MHA12 el que presenta mayor R2 ajustado es MHA12_5, con un valor de 0.734. Este valor es peor que los obtenidos con M2, M2int, Mtodas y MHA. Con lo cual no incluiremos el modelo en la tabla comparativa ni en el estudio posterior.

2.9 Puedes añadir cualquier otro análisis que consideres de interés.

Recopilando los resultados obtenidos en la tabla comparativa

```
print(knitr::kable(round(comp, 3), format = "pandoc",
                   col.names = c("M1", "M2", "M2int", "Mtodas", "MHA"), align='c'))
```

```
##
##
##                            M1       M2      M2int    Mtodas      MHA
## ------------------------  -------  -------  -------  --------  -------
## R2                         0.434    0.759    0.761    0.844     0.843
## R2 ajustado                0.433    0.759    0.760    0.841     0.840
## Residual standard error    1.541    1.005    1.003    0.817     0.818
```

MHA por ahora MHA es el modelo con mejores características de ajuste a los datos. Lo obtuvimos realizando la regresión paso a paso hacia adelante, veamos si mejora los resultados iterar en ambas direcciones.

```
MHA2=stepAIC(null, scope=list(upper=full), direction="both", criterion=AIC, trace=0)
summary(MHA2)
```

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

```
## lm(formula = solTrainY ~ MolWeight + SurfaceArea1 + NumNonHAtoms +
##     NumHydrogen + NumAtoms + NumRings + NumOxygen + NumNitrogen +
##     NumDblBonds + NumMultBonds + NumAromaticBonds + NumNonHBonds +
##     NumChlorine + HydrophilicFactor + NumRotBonds, data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2765 -0.4829  0.0411  0.5092  3.2373
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        9.47482    1.70744   5.549 3.74e-08 ***
## MolWeight         -1.60275    0.19542  -8.202 7.79e-16 ***
## SurfaceArea1       0.16448    0.02525   6.514 1.19e-10 ***
## NumNonHAtoms      10.07915    2.57347   3.917 9.64e-05 ***
## NumHydrogen        1.39731    0.14685   9.515  < 2e-16 ***
## NumAtoms          -6.24865    0.71131  -8.785  < 2e-16 ***
## NumRings           1.58741    0.40006   3.968 7.80e-05 ***
## NumOxygen          1.15891    0.13600   8.521  < 2e-16 ***
## NumNitrogen        1.60173    0.17314   9.251  < 2e-16 ***
## NumDblBonds       -0.54574    0.14380  -3.795 0.000157 ***
## NumMultBonds      -0.54470    0.08857  -6.150 1.15e-09 ***
## NumAromaticBonds   0.46388    0.11658   3.979 7.45e-05 ***
## NumNonHBonds      -5.11995    1.39967  -3.658 0.000268 ***
## NumChlorine       -0.62628    0.21604  -2.899 0.003832 **
## HydrophilicFactor -0.15336    0.05475  -2.801 0.005203 **
## NumRotBonds       -0.17023    0.08711  -1.954 0.050983 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8176 on 935 degrees of freedom
## Multiple R-squared:  0.8429, Adjusted R-squared:  0.8404
## F-statistic: 334.5 on 15 and 935 DF,  p-value: < 2.2e-16
```

Vemos que el modelo resultante MHA2 tiene valores de R2 y RSE casi identicos a los obtenidos para MHA. El modelo MHA2 contiene las mismas variables regresoras que MHA excepto NumCarbon, que tal y como ya obtuvimos no era significativa y se podía eliminar.

El modelo MHA tiene 15 variables regresoras: HydrophilicFactor, MolWeight, NumAromaticBonds, NumAtoms, NumChlorine, NumDblBonds, NumHydrogen, NumMultBonds, NumNitrogen, NumNonHAtoms, NumNonHBonds, NumOxygen, NumRings, NumRotBonds, SurfaceArea1

En el análisis de multicolinealidad vimos que las siguientes variables regresoras tenían problema de multicolinealidad:

MolWeight, NumAromaticBonds, NumAtoms, NumBonds, NumCarbon, NumHydrogen, NumMultBonds, NumNonHAtoms, NumNonHBonds, NumOxygen, NumRings, SurfaceArea1, SurfaceArea2

Con lo cual, vamos a eliminar las variables del modelo MHA que presentan problemas de multicolinealidad y veremos si obtenemos mejores resultados con el nuevo modelo simplificado.

Por tanto, construiremos un nuevo modelo MHA3 con las siguientes variables explicativas: HydrophilicFactor, NumChlorine, NumDblBonds, NumNitrogen, NumNonHBonds y NumRotBonds

```
MHA3=lm(solTrainY ~ HydrophilicFactor + NumChlorine + NumDblBonds +
          NumNitrogen + NumNonHBonds + NumRotBonds, data = EntrenamientoXY)
summary(MHA3)
```

```
##
## Call:
## lm(formula = solTrainY ~ HydrophilicFactor + NumChlorine + NumDblBonds +
##     NumNitrogen + NumNonHBonds + NumRotBonds, data = EntrenamientoXY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1644 -0.7617  0.0904  0.7648  3.8487
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        2.67873    0.15433  17.357  < 2e-16 ***
## HydrophilicFactor  0.64499    0.04216  15.299  < 2e-16 ***
## NumChlorine       -3.47959    0.22383 -15.546  < 2e-16 ***
## NumDblBonds        0.65197    0.11745   5.551 3.69e-08 ***
## NumNitrogen        1.05453    0.16311   6.465 1.62e-10 ***
## NumNonHBonds      -1.55212    0.05057 -30.694  < 2e-16 ***
## NumRotBonds       -0.07889    0.05918  -1.333    0.183
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.18 on 944 degrees of freedom
## Multiple R-squared:  0.6696, Adjusted R-squared:  0.6675
## F-statistic: 318.8 on 6 and 944 DF,  p-value: < 2.2e-16
```

Observamos que en este caso el R2 es mucho mas pequeño que para el resto de modelos, esto es debido a que hemos eliminado variables regresoras importantes en el modelo. Sería necesario realizar un estudio de colienalidad entre todas las variables regresoras y eliminar solo aquellas que no estén explicadas por el resto.

También podríamos estudiar si la inclusión de las variables FP eliminadas del conjunto de entrenamiento están haciendo que obtengamos peores resultados.

Otro estudio que tambien podría ser interesante abordar es el de los valores outlier. Para determinar aquellos valores que estén desviando los resultados y que sea bueno limpiar de los datos observados para obtener mejores ajustes y predicciones.

3 Cuestiones sobre el conjunto TestXY

3.1 Construye el fichero "TestXY" de la siguiente forma:

(a) Constuye un fichero "TestX" eliminando de "solTestX" todas las variables "FP..." que ocupan las 208 primeras posiciones.

(b) Construye el fichero "TestXY" añadiendo a "TestX" la variable "solTestY".

Construiré el conjunto de test con las variables transformadas, puesto que para entrenar usé las variables transformadas, de otro modo los resultados que obtenga no serán correctos.

```
TestX<-data.frame(solTestXtrans[209:ncol(solTestXtrans)])
TestXY<-data.frame(TestX, solTestY)
str(TestXY)
```

```
## 'data.frame':    316 obs. of  21 variables:
##  $ MolWeight        : num  4.56 4.5 4.71 4.62 4.81 ...
##  $ NumAtoms         : num  2.2 2.64 2.71 3 2.77 ...
```

```
##  $ NumNonHAtoms     : num  1.79 1.95 2.2 2.08 2.3 ...
##  $ NumBonds         : num  2.08 2.56 2.71 3 2.77 ...
##  $ NumNonHBonds     : num  1.9 2.15 2.76 2.58 2.92 ...
##  $ NumMultBonds     : num  0.799 0.799 2.945 0 3.243 ...
##  $ NumRotBonds      : num  0 1.099 0 0 0.693 ...
##  $ NumDblBonds      : num  0.567 0.567 0 0 0.567 ...
##  $ NumAromaticBonds : num  0 0 1.95 0 1.95 ...
##  $ NumHydrogen      : num  1.72 2.89 2.64 3.86 2.64 ...
##  $ NumCarbon        : num  1.3 1.72 2.64 2.37 2.64 ...
##  $ NumNitrogen      : num  0 0.457 0 0.585 0.585 ...
##  $ NumOxygen        : num  1.099 1.099 1.099 0 0.693 ...
##  $ NumSulfer        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NumChlorine      : num  0.375 0 0 0 0 0 0 0 0 0 ...
##  $ NumHalogen       : num  0.375 0 0 0 0 0 0 0 0 0 ...
##  $ NumRings         : num  0 0 0.693 0.693 0.693 ...
##  $ HydrophilicFactor: num  0.417 0.915 0.652 0.734 0.65 ...
##  $ SurfaceArea1     : num  8.25 9.77 8.59 6.57 10.1 ...
##  $ SurfaceArea2     : num  8.25 9.77 8.59 6.57 10.1 ...
##  $ solTestY         : num  0.93 0.85 0.81 0.74 0.61 0.58 0.57 0.56 0.52 0.45 ...
```

3.2 Calcula los valores ajustados por el modelo MHA12 para todos los valores de los regresores del conjunto test

```
# Valores ajustados por el modelo MHA12 sobre conjunto test
solTestY_gorro=predict(MHA12, TestX)
head(solTestY_gorro)
```

```
##          20          21          23          25          28          31
## -0.02612996  0.57392244 -0.48062648 -0.66566582 -0.46183532  1.22857316
```

```
# Valores observados y predicciones
lmValues=data.frame(obs=solTestY, pred=solTestY_gorro)
head(lmValues)
```

```
##     obs        pred
## 20 0.93 -0.02612996
## 21 0.85  0.57392244
## 23 0.81 -0.48062648
## 25 0.74 -0.66566582
## 28 0.61 -0.46183532
## 31 0.58  1.22857316
```

```
defaultSummary(lmValues)
```

```
##      RMSE  Rsquared
## 1.0256588 0.7592514
```

3.3 Calcula los residuos correspondientes al modelo MHA12 para todos los valores de los regresores del conjunto test

```
residuos=solTestY-solTestY_gorro
head(residuos)
```

```
##         20         21         23         25         28         31
##  0.9561300  0.2760776  1.2906265  1.4056658  1.0718353 -0.6485732
```

3.4 Calcula el RSE resultante de aplicar el modelo MHA12 sobre el conjunto test.

```
(RSE_test=sqrt(sum((TestXY$solTestY-solTestY_gorro)^2)/(nrow(TestXY)-2)))
```

```
## [1] 1.02892
```

3.5 Compara el RSE sobre el conjunto entrenamiento y el test. Comenta los resultados.

```
(RSE_entrenamiento=summary(MHA12)$sigma)
```

```
## [1] 1.077333
```

En el caso del modelo MHA12, los resultados sobre el conjunto test son mejores que sobre el conjunto de entrenamiento, el RSE es algo menor en el caso del conjunto de test.

3.6. Puedes añadir cualquier otro análisis que consideres de interés.

Compararemos a continuacion el valor de RSE en el conjunto test y en el conjunto entrenamiento para el modelo MHA

```
solTestY_gorro_MHA=predict(MHA, TestX)
(RSE_test_MHA=sqrt(sum((TestXY$solTestY-solTestY_gorro)^2)/(nrow(TestXY)-2)))
```

```
## [1] 1.02892
```

```
(RSE_entrenamiento_MHA=summary(MHA)$sigma)
```

```
## [1] 0.81804
```

Vemos que el error cometido al estimar sobre el conjunto de test es igual para el modelo MHA y MHA12. Sin embargo obteníamos mejores ajustes con MHA que con MHA12.

A continuación compararé los resultados obtenidos los resultantes de realizar cross validacion con 300 folds

```
set.seed(100)
ctrl=trainControl(method = "cv", number=300, selectionFunction = "best")
lm_cross=train(x=EntrenamientoX, y=solTrainY, method = "lm", trControl =ctrl )
lm_cross$results
```

```
##   intercept      RMSE  Rsquared    RMSESD RsquaredSD
## 1      TRUE 0.7399781 0.8690686 0.3695659  0.2206664
```

Vemos que los resultados son mejores con el modelo obtenido con cross validación.