Trabajo 3: AJUSTE DE MODELOS LINEALES

Nuria Rodríguez Barroso, Juan Luis Suárez Díaz.

27 de mayo de 2017

Contents

Clasificación.

Comprensión del problema a resolver.

Para el problema de clasificación hemos elegido la base de datos de *South African Heart Disease*, que almacena una muestra recapituladora de hombres en alto riesgo de cardiopatía en la región de Western Cape, en Sudáfrica. Hay dos casos de CHD (0 o 1). Algunos de los hombres que tienen CHD positivo, se han sometido a un tratamiento de reducción de la presión sanguínea, siendo los datos que aparecen en la muestra posteriores a estos tratamientos. Estos datos datan de 1983.

Nuestra base de datos consta de 462 muestras donde cada una de ellas cuenta con 10 atributos, uno de ellos la variable de respuesta. Los diferentes atributos a tratar son:

- **sbp:** presión arterial sistólica. Toma valores entre 101 y 218, siendo la media 138.3.
- **tobacco:** tabaco acumulativo (en kg). Toma como valor mínimo 0 y máximo 31.2. En este caso la media es de 3.6356.
- **ldl:** lipoproteína de baja densidad (colesterol). Toma valores entre 0.98 y 15.330, siendo la media 4.74.
- **adiposidad:** Tomando valores entre 6.74 y 42.49, siendo la media de los valores 25.41.
- **famhist:** historial familiar de cardiopatías. Toma como valores {Present, Absent}, habiendo del primer tipo 270 y del segundo 192.
- **typea:** Personalidad Tipo-A (mide el grado de estrés en el día a día). Toma valores entre 13 y 78, siendo la media 53.1.
- **obesity:** Obesidad. Toma valores entre 14.7 y 46.58, siendo la media 26.04.
- **alcohol:** Actual consumición de alcohol. Toma valores entre 0 y 147.19, encontrándose el valor medio en 17.04.
- **age:** Edad de los hombres al comienzo de las pruebas. Se encuentra entre 15 y 64 años, siendo la edad media 42.82.
- **chd:** Variable de respuesta, indica si se tiene o no alguna cardiopatía. Toma como valores {0,1}, siendo la media 0.3463.

Preprocesado de datos.

Como podemos observar en el breve estudio de los diferentes atributos que vamos a trabajar, cada uno toma valores en una franja muy diferente, lo que hace primar unos atributos sobre otros en los métodos basados en distancias. Además, algunos atributos presentan una gran asimetría, lo cual también es conveniente evitar. Por estos motivos y otros más que estudiaremos a continuación, tenemos que preprocesar los datos.

Modificación de los atributos cualitativos.

Tenemos que convertir los atributos cualitativos en atributos numéricos para que las funciones que usemos más adelante puedan trabajar con ellos. En nuestro problema concreto, solo contamos con un atributo cualitativo: *famhist*, que toma los valores {Present, Absent}. Convertimos este atributo en el atributo *present_famhist*, que tomará el valor 1, cuando *famhist* tomaba el valor Present y 0 en el otro caso. Así, el atributo *present_famhist* tomará valores en {0,1}.

Para el resto de pasos, utilizaremos una función llamada *preProcess()*, la cual terminará con el preprocesado de los datos. Esta función realizará los siguientes cambios:

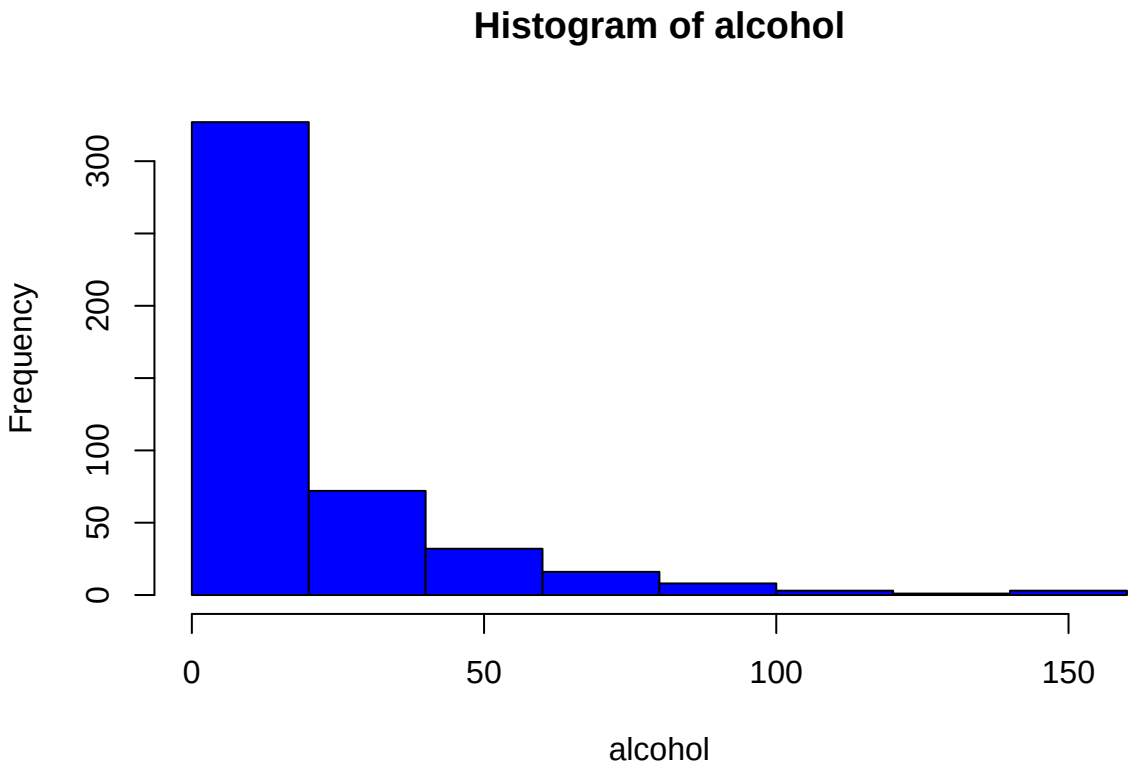
Tratamiento de la asimetría con BoxCox.

Como ya hemos comentado anteriormente, hay varios atributos que presentan una alta asimetría, lo cual podría hacer que los métodos de predicción que apliquemos a continuación obtengan resultados peores. Para solucionar esto, utilizaremos un método llamado BoxCox. Este método se basa en la transformación potencial

según un valor (λ) para aumentar la correlación entre las variables. Para elegir la mejor potencia (mejor λ), se busca entre los λ que proporcionen un menor error residual. Aunque en la práctica, esto se realizará de manera automátcica con la función *preProcess()* vamos a ver cómo funciona en el caso de un atributo. Para que se vea mejor el funcionamiento, vamos a elegir el atributo que presente una mayor asimetría. Para ello, ordenamos los atributos en función de su asimetría:

```
##         alcohol         tobacco             ldl             sbp
##       2.2977031       2.0657278       1.3045896       1.1729355
##         obesity             chd             age           typea
##       0.8993498       0.6438924       0.3792590       0.3441914
## present_famhist       adiposity
##       0.3414684       0.2132541
```

Si dibujamos el histograma correspondiente al atributo con mayor asimetría, *alcohol* corroboramos que los datos se encuentran muy concentrados en los primeros valores que este atributo toma.



Vamos a aplicar ahora el método *BoxCox* y veremos cómo mejora la simetría del atributo.

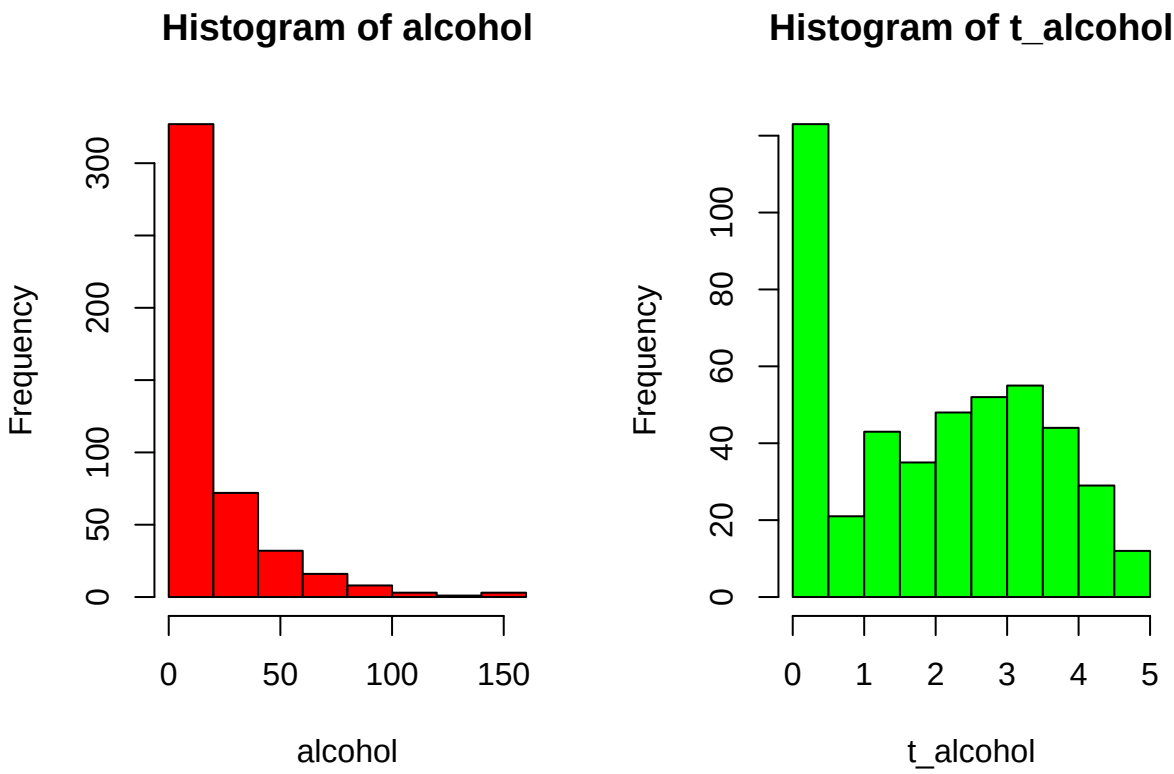
```
## Box-Cox Transformation
##
## 462 data points used to estimate Lambda
##
## Input data summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.51    7.51   17.04   23.89  147.19
##
## Lambda could not be estimated; no transformation is applied
```

Como observamos, no se encuentra un λ válido para realizar la transformación, y esto se debe a que entre los

valores que toma el atributo se encuentra el 0, punto en el cual no está definida la función logaritmo. Para solucionar esto, realizamos una translación de los datos de la forma: datos = min(datos) + 1 + datos, para así conseguir que el mínimo de los datos se desplace a 1. Tras realizar esta transformación obtenemos:

```
## [1] "La asimetría del alcohol transformado es:"
```

```
## [1] 0.02949713
```



Para aplicar esta transformación a todos los atributos con el λ correspondiente, pasaremos como parámetro al método *preProcess* que realice el método *BoxCox* y todo esto se hará de forma automática.

Eliminación de atributos con PCA.

El algoritmo PCA (Principal Components Analysis) es un filtro no supervisado que es de gran utilidad cuando disponemos de una base de datos con un gran número de atributos, entre los que algunos pueden ser redundantes o irrelevantes. Como nuestra base de datos solo consta de 10 atributos, no es necesaria la aplicación de este método.

Centrar y escalar.

También es conveniente centrar y escalar las variables para que no prioricen unas sobre otras, dado que puede ser útil en algunos métodos que utilizaremos más adelante. Al escalar una variable lo que se está haciendo es dividir cada dato entre la desviación típica del conjunto de datos, transformando así el conjunto de datos en un conjunto de varianza 1. En cuanto a centrar el conjunto de datos, lo que se hace es restar a cada dato la media del conjunto, transformándolo así en un conjunto de media 0. Al escalar y centrar a la vez, estamos aplicando la transformación $X \leftarrow (X - \mu)/\sigma$, normalizando así el conjunto a un conjunto de

media 0 y varianza 1. Esto nos permite tener los atributos normalizados, y además manteniendo las mismas distribuciones entre los distintos atributos, como veremos más adelante en la regresión. El método *preProcess* se encargará de centrar las variables en 0 y de escalar las variables para tener varianza unitaria. Para ello, bastaría con pasar como argumento *scale* y *center* cuando llamemos al método *preProcess*.

Llamada al método PreProcess.

Una vez entendidas las modificaciones que vamos a realizar a los datos, utilizaremos el método preProcess que se encarga de realizar todas estas modificaciones sobre el conjunto de datos pasado como argumento. Como ya hemos comentado, no vamos a aplicar el método *PCA*, luego la llamada quedaría de la siguiente forma:

```
ObjetoTrans = preProcess(sahd[,names(sahd)!="chd"],method = c("BoxCox","center","scale"))
sahdTrans <- predict(ObjetoTrans,sahd)
```

Conjuntos de validación, training y test usados.

A continuación pasaremos a explorar los distintos modelos sobre los que resolver el problema de clasificación para nuestro conjunto de datos. El procedimiento de validación que usaremos consistirá en tomar múltiples veces distintos conjuntos de entrenamiento sobre nuestro conjunto de datos (una vez transformados), con los que aprenderemos el modelo. Usaremos el resto del conjunto como test para evaluar cómo de bien predice el modelo aprendido con nuevos datos. Las proporciones utilizadas serán del 70 % de los datos para train, y el 30 % para test.

Selección de clases de funciones a usar

Los modelos que vamos a intentar ajustar son los proporcionados por la función `glm` (Generalized Linear Models) de `R`. Para cada familia, siempre que admitan, utilizaremos distintas funciones de enlace. Las funciones de enlace nos permiten establecer una relación entre la media de la respuesta y los predictores del modelo. Las familias que vamos a considerar para el ajuste son:

- **Binomial**, con link **logit**. Regresión logística.
- **Binomial**, con link **probit**. Modelo binomial, con función de enlace $\Phi^{-1}(\mu)$, donde Φ es la distribución acumulada de la distribución normal.
- **Binomial**, con link **cauchit**. Modelo binomial, cuya función de enlace es la análoga a la del modelo anterior sobre una distribución de Cauhy, en lugar de la normal.
- **Gaussiana**, con link **identity**. Regresión lineal.
- **Gaussiana**, con link **log**. Distribución normal con función de enlace logarítmica.
- **Poisson**. Distribución de Poisson.
- **Quasi**. Este modelo no tiene una varianza determinada como en el resto de familias. Indicaremos la especificación de varianza `"constant"`
- **Quasibinomial.** Distribución binomial, con la única diferencia de que no fija el parámetro de dispersión (intenta describir varianza adicional en los datos que no puede ser explicada mediant una distribución binomial).
- **Quasipoisson.** Distribución de Poisson, con la única diferencia de que no fija el parámetro de dispersion.

Una vez definidos los modelos y las funciones a usar, procedemos al ajuste de los distintos modelos y al análisis de sus errores:

```
##                            Ein      Eout
## Binomial - Logit    0.2537771 0.2774101
## Binomial - Probit   0.2548916 0.2771942
## Binomial - Cauchit  0.2466873 0.2788489
## Gaussian - Identity 0.2536533 0.2775540
```

```
## Gaussian - Log      0.2473994 0.2689928
## Poisson             0.2483591 0.2658273
## Quasi               0.2536533 0.2775540
## Quasibinomial       0.2537771 0.2774101
## Quasipoisson        0.2483591 0.2658273
```

Obtenemos que el modelo que mejores resultados proporciona es el de Poisson. Hay que destacar que los resultados de Poisson coinciden con los de quasipoisson, pero elegimos el de Poisson por ser más conocido.

Regularización

A continuación nos planteamos la necesidad de regularización, sobre el mejor modelo que hemos obtenido. Para ello utilizamos la regularización lasso (least absolute shrinkage and selection operator). Lasso es un método que lleva a cabo la regularización a la misma vez que realiza selección de características.

El objetivo se basa en reducir el error de predicción, para ello, se ocupa de reducir la función:

$$R(\beta) = \sum_{i=1}^{n}(y_i - x_i\omega)^2 + \lambda \sum_{i=1}^{p} |\omega_i|$$

donde n es el número de muestras y p el número de atributos y w el vector de pesos solución. Así, obtiene diferentes valores de λ. Entre estos valores devueltos, podemos considerar dos de ellos:

- *lambda.min*, que nos devuelve el valor del λ para el que se minimiza el error obtenido.
- *lambda.1se*. Aunque el λ anterior sea menor, presenta una mayor dependencia de las particiones seleccionadas en la validación cruzada. La regla `1se` (one standard error), elige un valor de λ para lo suficiente cercano a `lambda.min` para el cual el modelo obtenido sea lo suficiente simple, reduciendo así la dependencia del error respecto a las validaciones.

Podemos contemplar cualquiera de estos valores de λ para regularizar nuestro modelo.

Para contestar a la pregunta de si era necesario aplicar regularización a nuestra base de datos, realizamos 100 experimentos en los que, para diferentes subconjuntos de datos de nuestra muestra calculamos el error al regularizar con ambos valores de λ y al no regularizar. Debemos quedarnos con el modelo que menos error presente.

```
## Etest con lambda min:  0.2702878
```

```
## Etest con lambda 1se:  0.2828058
```

```
## Etest con el modelo original:  0.2673381
```

Como podemos observar, el error medio obtenido es menor con el modelo sin regularizar, obteniendo así una respuesta negativa a la pregunta. Por tanto, seguiremos con el modelo sin regularización.

A modo de ampliación, comentar que el método de regularización lasso, cuando devuelve los coeficientes nulos significa que está despreciando estos atributos para la predicción. Si imprimimos los coeficientes correspondientes al valor de *lambda.1se* observamos que los atriburos que no selecciona para la predicción son: sbp, adiposity, typea, obesity y alcohol.

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)     -1.10346302
## sbp              .
## tobacco          0.06870336
## ldl              0.09622207
## adiposity        .
## present_famhist  0.12878139
```

```
## typea            0.03107499
## obesity          .
## alcohol          .
## age              0.29203487
```

A continuación, en la selección del número de atributos a utilizar, veremos que dichos atributos son, en efecto, algunos de los que participan en menos combinaciones "óptimas", por lo tanto, serán algunos de los menos relevantes.

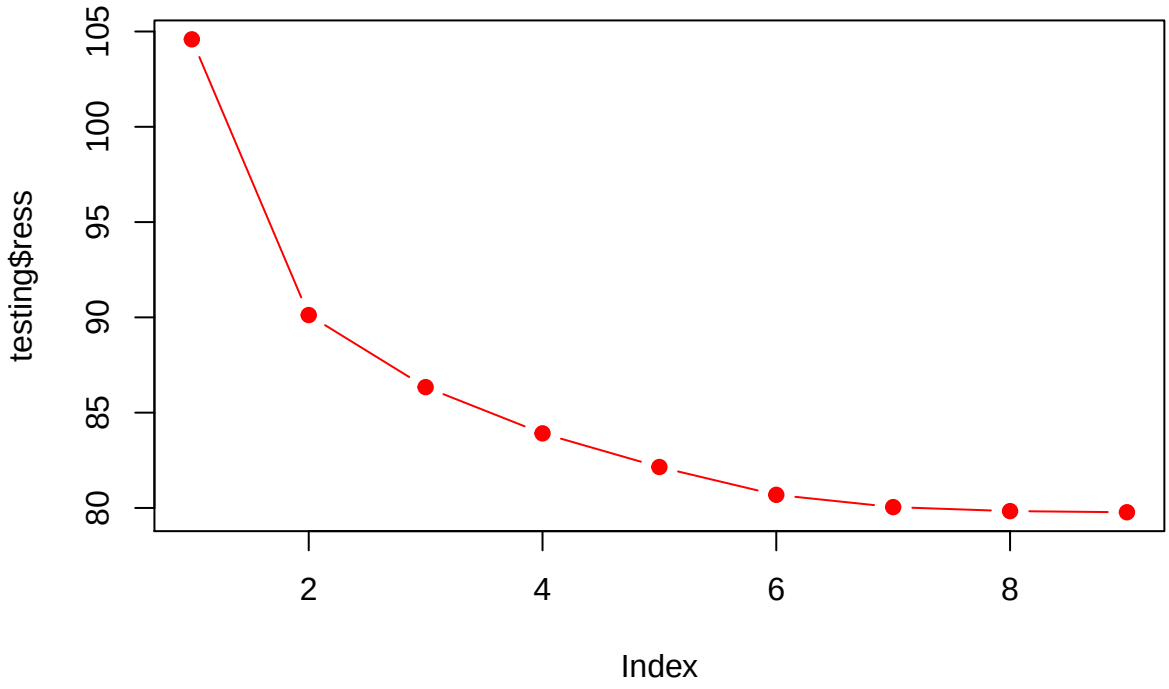
Optimización del número de atributos para el modelo seleccionado

Para optimizar el número de atributos para el modelo seleccionado, vamos a hacer uso de la función llamada *regsubsets*, esta función junto con las opciones de *method = "exhaustive* y *nbest = 1* realizará una búsqueda exhaustiva del mejor atributo (el que produce menor error cuadrático), la mejor pareja de atributos, el mejor trío, etcétera.

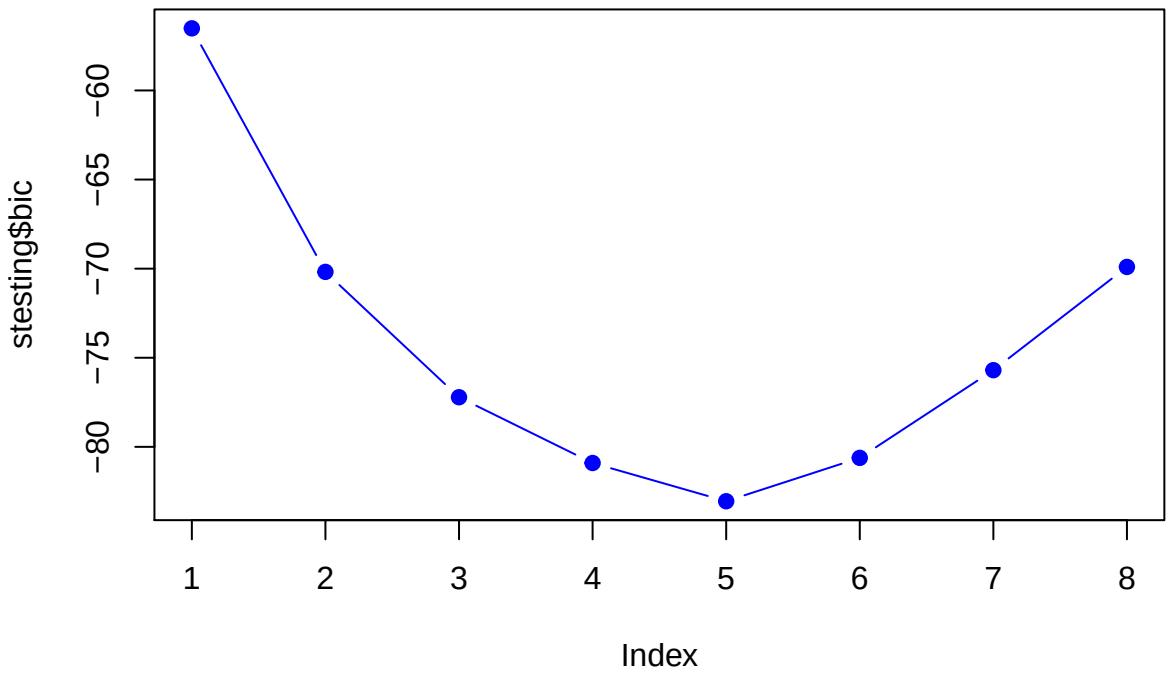
El método nos proporciona el siguiente esquema en el que podemos apreciar las combinaciones de atributos elegidos.

```
##          sbp tobacco ldl adiposity present_famhist typea obesity alcohol
## 1  ( 1 ) " " " "     " " " "       " "              " "   " "     " "
## 2  ( 1 ) " " " "     " " " "       "*"              " "   " "     " "
## 3  ( 1 ) " " "*"     " " " "       "*"              " "   " "     " "
## 4  ( 1 ) " " "*"     " " " "       "*"              "*"   " "     " "
## 5  ( 1 ) " " "*"     "*" " "       "*"              "*"   " "     " "
## 6  ( 1 ) " " "*"     "*" " "       "*"              "*"   "*"     " "
## 7  ( 1 ) " " "*"     "*" "*"       "*"              "*"   "*"     " "
## 8  ( 1 ) "*" "*"     "*" "*"       "*"              "*"   "*"     " "
##          age
## 1  ( 1 ) "*"
## 2  ( 1 ) "*"
## 3  ( 1 ) "*"
## 4  ( 1 ) "*"
## 5  ( 1 ) "*"
## 6  ( 1 ) "*"
## 7  ( 1 ) "*"
## 8  ( 1 ) "*"
```

En este punto nos planteamos cuántos atributos utilizar para nuestro modelo. Claramente, cuantos más atributos utilicemos menor será el error producido en la muestra. Esto lo podemos corroborar dibujando la gráfica del error por mínimos cuadrados en función del número de atributos elegidos.



Sin embargo, debemos de plantearnos qué número de atributos sería el óptimo si penalizáramos también el número de atributos utilizados. Es decir, cuándo una función que combine el error producido junto con el número de atributos utilizado se minimiza. Para ello utilizamos la función *BIC*, cuya gráfica en función del número de atributos podemos observar a continuación:



El criterio BIC (Bayesian Information Criterion) se basa en seleccionar el modelo con menor función BIC asociada. La función BIC es una función que depende logarítmicamente del tamaño de la muestra n, el número de atributos k y el estimador máximo verosímil, $\hat{L}$. Concretamente, $BIC = \ln(n)k - 2\ln(\hat{L})$ Dicha función alcanza un mínimo con 5 atributos, por tanto, este será el número de atributos que utilizaremos para nuestro modelo.

Transformación de atributos

A continuación, nos planteamos la búsqueda de una transformación polinómica de los atributos para la cual nuestro modelo tenga una mayor capacidad de aprender y predecir nuestro conjunto de datos. Para ello utilizamos un algoritmo de búsqueda greedy. Para cada atributo, vamos eligiendo distintos exponentes y nos quedamos con el exponente que proporcione menor error de validación. Cuando llegamos al siguiente atributo, aplicamos el mismo procedimiento, manteniendo para los atributos anteriores el mejor exponente encontrado. Además, como cuando dos modelos proporcionan resultados similares siempre es mejor quedarse con el más simple, fijaremos una tolerancia para la cual, si no hay mejoras significativas en la nueva transformación, nos quedemos con la transformación mejor obtenida previamente, que tendrá un exponente menor y por tanto será más simple el ajuste.

Aplicamos el algoritmo. Los resultados obtenidos son:

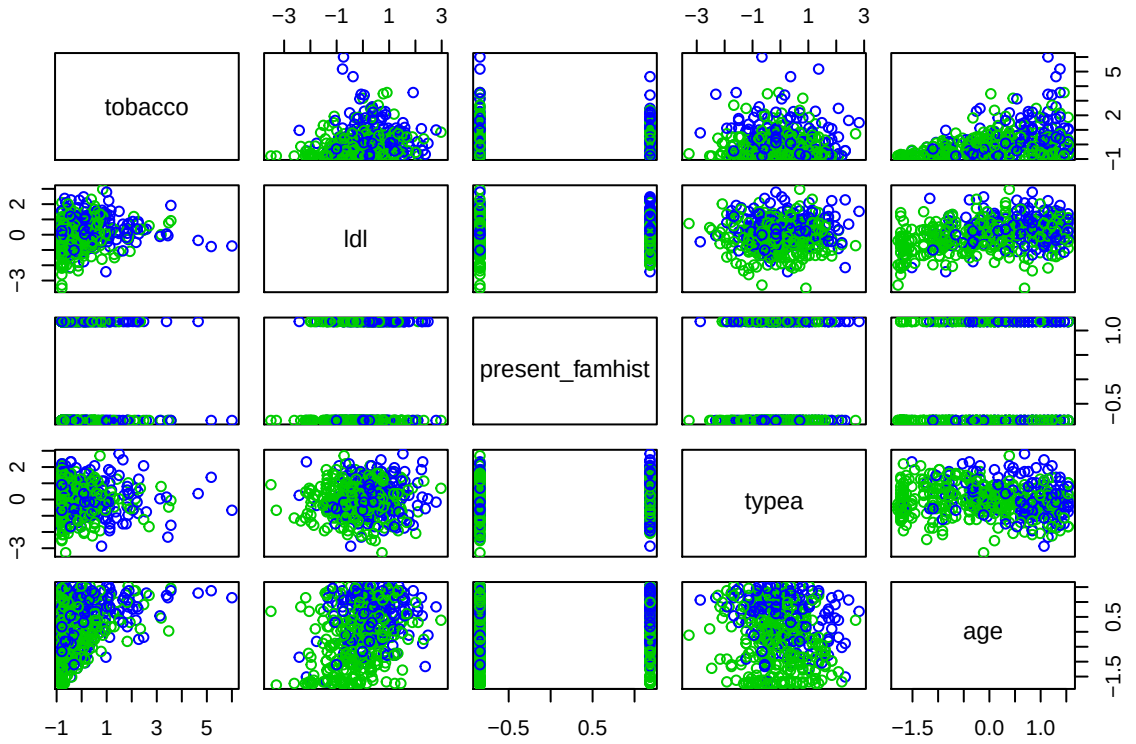
```
## Attr =  1 , Exp =  1 ,  Ein =  0.2509907 Eout =  0.2653237
## Attr =  1 , Exp =  2 ,  Ein =  0.2604334 Eout =  0.2658273
## Attr =  1 , Exp =  3 ,  Ein =  0.261517 Eout =  0.2693525
## Attr =  1 , Exp =  4 ,  Ein =  0.2627554 Eout =  0.2689209
## Attr =  1 , Exp =  5 ,  Ein =  0.2624458 Eout =  0.2690647
## Attr =  1 , Exp =  6 ,  Ein =  0.2586687 Eout =  0.2784173
```

```
## Attr =  2 , Exp =  1 ,  Ein =  0.2504025 Eout =  0.263741
## Attr =  2 , Exp =  2 ,  Ein =  0.2642105 Eout =  0.2807914
## Attr =  2 , Exp =  3 ,  Ein =  0.2611765 Eout =  0.2672662
## Attr =  2 , Exp =  4 ,  Ein =  0.2674303 Eout =  0.283741
## Attr =  2 , Exp =  5 ,  Ein =  0.26387 Eout =  0.2763309
## Attr =  2 , Exp =  6 ,  Ein =  0.2731269 Eout =  0.2805755
## Attr =  3 , Exp =  1 ,  Ein =  0.2505573 Eout =  0.265036
## Attr =  3 , Exp =  2 ,  Ein =  0.2531889 Eout =  0.2630216
## Attr =  3 , Exp =  3 ,  Ein =  0.2495666 Eout =  0.2660432
## Attr =  3 , Exp =  4 ,  Ein =  0.2515789 Eout =  0.2629496
## Attr =  3 , Exp =  5 ,  Ein =  0.2525697 Eout =  0.2616547
## Attr =  3 , Exp =  6 ,  Ein =  0.2505573 Eout =  0.2638129
## Attr =  4 , Exp =  1 ,  Ein =  0.2511455 Eout =  0.2641727
## Attr =  4 , Exp =  2 ,  Ein =  0.2547059 Eout =  0.2590647
## Attr =  4 , Exp =  3 ,  Ein =  0.2527554 Eout =  0.2670504
## Attr =  4 , Exp =  4 ,  Ein =  0.2539319 Eout =  0.2684892
## Attr =  4 , Exp =  5 ,  Ein =  0.2545201 Eout =  0.2673381
## Attr =  4 , Exp =  6 ,  Ein =  0.2549845 Eout =  0.2648201
## Attr =  5 , Exp =  1 ,  Ein =  0.2513313 Eout =  0.2640288
## Attr =  5 , Exp =  2 ,  Ein =  0.2691641 Eout =  0.2874101
## Attr =  5 , Exp =  3 ,  Ein =  0.2549226 Eout =  0.2722302
## Attr =  5 , Exp =  4 ,  Ein =  0.2762848 Eout =  0.2814388
## Attr =  5 , Exp =  5 ,  Ein =  0.2599381 Eout =  0.2671223
## Attr =  5 , Exp =  6 ,  Ein =  0.2756037 Eout =  0.2848921

## Vector de exponentes:  1 1 1 1 1

## Eout estimado:  0.2640288
```

Vemos que, para exponentes de hasta tamaño 6 no se aprecian mejoras significativas con respecto a los coeficientes lineales iniciales.

Conclusiones.

Por tanto, el modelo óptimo escogido es un modelo lineal que utiliza los atributos tobacco, ldl, present_famhist, typea y age. A este conjunto de datos le aplicamos una regresión de Poisson obteniendo un error estimado del 26.46763%. Es decir, tras todos los modelos probados el error no ha conseguido disminuir del 25 %, lo que nos indica que los datos no son separables en gran medida. Esto se puede ilustrar en la siguiente gráfica, en la que se muestran los atributos finales escogidos comparados dos a dos, junto con la clase que determina el dato en cuestión. Vemos que en general, es difícil apreciar patrones de separación (aunque esto no nos asegura nada, puesto que la dimensión del número de atributos es mayor, y comparándolos 2 a 2 no se puede afirmar nada sobre la separabilidad real del conjunto).



En la gráfica anterior podemos observar además que hay atributos que separan las clases con mayor facilidad, por ejemplo, `age` y `ldl`. Podemos probar a clasificar los datos usando solo estos atributos, obteniendo los siguientes resultados:

```
## Ein =  0.3034056
```

```
## Etest =  0.323741
```

Es decir, de entre el par de atributos que mejor aparenta separar los datos, el error es bastante elevado, y al optimizar el número de atributos, aunque el error sigue siendo alto, conseguimos reducirlo bastante. Esto nos permite observar que el modelo ajustado, dentro de la poca separabilidad de los datos, es de calidad.

También hemos comprobado que, en general, el conjunto de datos es poco sensible a la regularización y a las transformaciones. Sobre esto último, al no haber diferencias relevantes con respecto a ninguna transformación polinómica, se mantiene la linealidad en los datos, puesto que entre modelos similares, siempre es mejor quedarnos con el de mayor simplicidad.

Regresión.

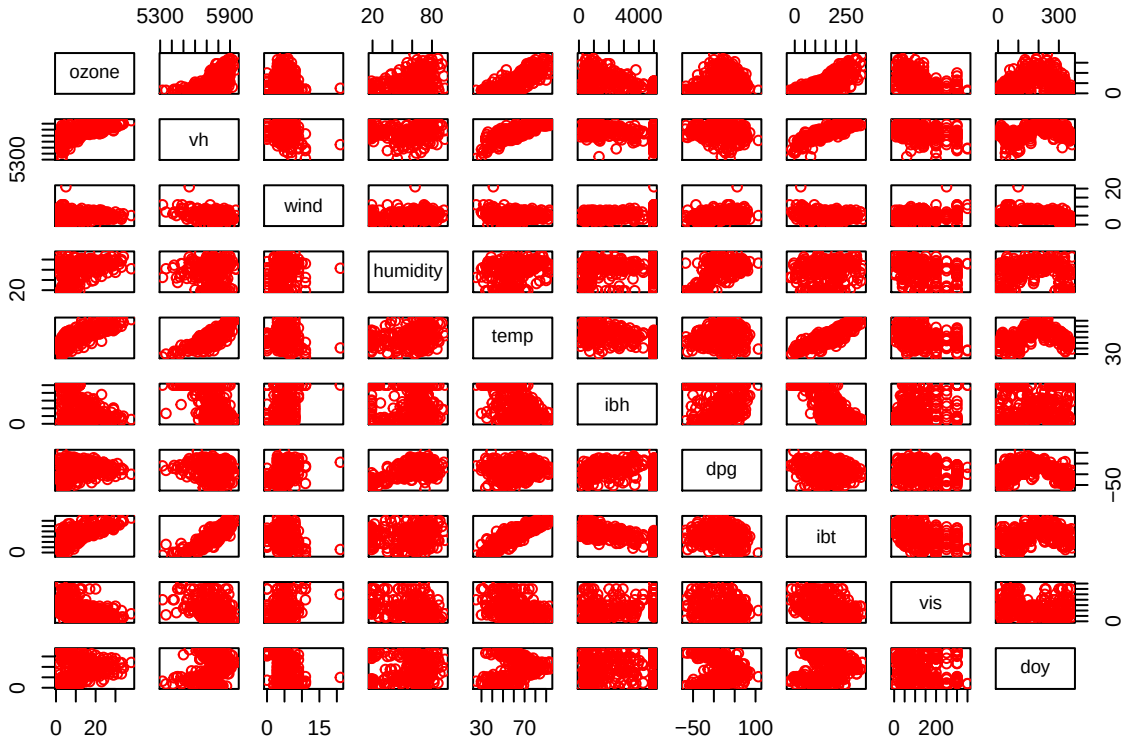
Para el problema de regresión hemos elegido la base de datos de *Los Ángeles Ozone*, la cual se centra en medir el nivel de concentración de ozono en la atmósfera. Para ello, se realizaban 8 mediciones hechas diariamente en Los Ángeles durante el año 1976. Aunque la idea era obtener el nivel de ozono para todos los días del año, algunos datos se han perdido así que no contiene todos los días del año (en concreto contiene 330 días). La base de datos consta de 10 atributos que son los siguientes:

- **ozone**: Es la variable de respuesta, mide la elevación máxima del ozono.
- **vh**: Vandenberg 500 mb Height

- **wind**: Velocidad del viento, medida en mph.
- **humidity:** Tanto por ciento de humedad.
- **temp:** Temperatura
- **ibh:** Altura sobre el nivel del mar (feet).
- **dpg:** Gradiente de presión de Daggot.
- **ibt:** Temperatura (ºF)
- **vis:** La visibilidad medida en millas.
- **day:** Día del año en el que se realizó la medición.

```
##      ozone             vh            wind           humidity
##  Min.   : 1.00   Min.   :5320   Min.   : 0.000   Min.   :19.00
##  1st Qu.: 5.00   1st Qu.:5690   1st Qu.: 3.000   1st Qu.:47.00
##  Median :10.00   Median :5760   Median : 5.000   Median :64.00
##  Mean   :11.78   Mean   :5750   Mean   : 4.891   Mean   :58.13
##  3rd Qu.:17.00   3rd Qu.:5830   3rd Qu.: 6.000   3rd Qu.:73.00
##  Max.   :38.00   Max.   :5950   Max.   :21.000   Max.   :93.00
##       temp            ibh              dpg              ibt
##  Min.   :25.00   Min.   : 111.0   Min.   :-69.00   Min.   :-25.0
##  1st Qu.:51.00   1st Qu.: 877.5   1st Qu.: -9.00   1st Qu.:107.0
##  Median :62.00   Median :2112.5   Median : 24.00   Median :167.5
##  Mean   :61.75   Mean   :2572.9   Mean   : 17.37   Mean   :161.2
##  3rd Qu.:72.00   3rd Qu.:5000.0   3rd Qu.: 44.75   3rd Qu.:214.0
##  Max.   :93.00   Max.   :5000.0   Max.   :107.00   Max.   :332.0
##       vis             doy
##  Min.   :  0.0   Min.   :  3.00
##  1st Qu.: 70.0   1st Qu.: 90.25
##  Median :120.0   Median :177.50
##  Mean   :124.5   Mean   :181.73
##  3rd Qu.:150.0   3rd Qu.:275.75
##  Max.   :350.0   Max.   :365.00
```

A continuación, se muestra una comparativa de todos los atributos 2 a 2 mediante gráficas, la cual nos puede ayudar más adelante a estimar el comportamiento no lineal de algunos atributos:



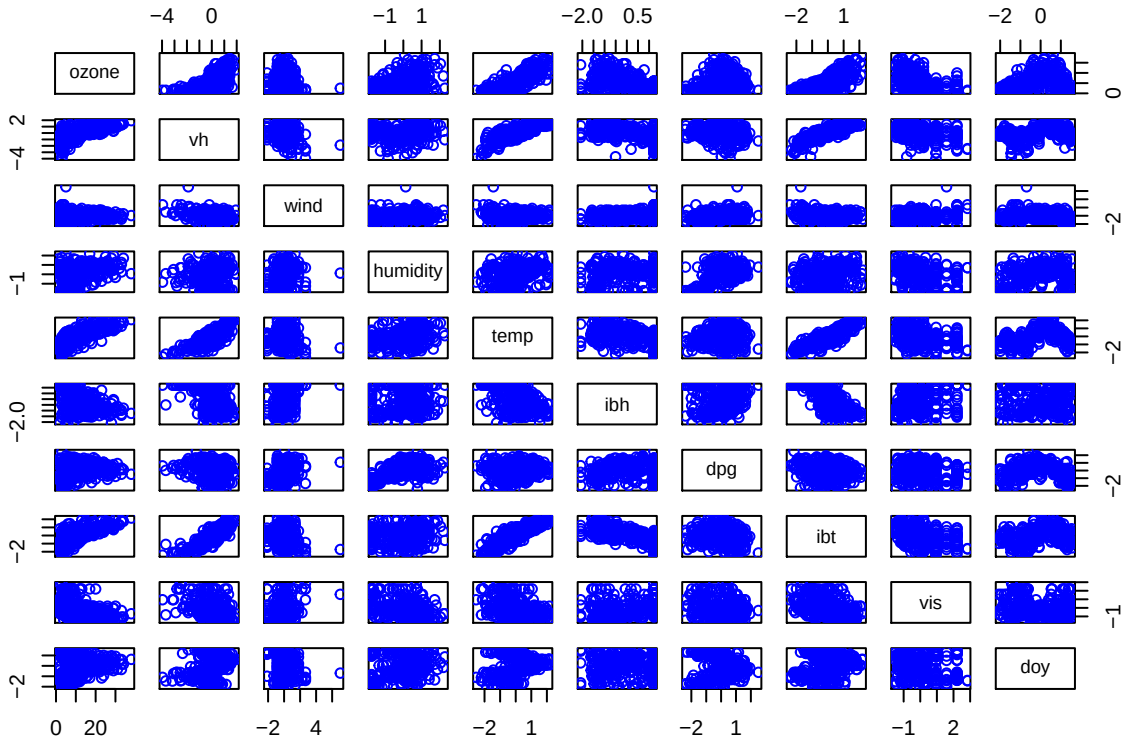
Realmente, las gráficas que más nos interesan son las de la primera fila, puesto que muestran `ozone`, el atributo sobre el que queremos hacer regresión, en función del resto de atributos. Aunque también, el resto de gráficas nos puede permitir intuir dependencias entre atributos, lo que nos puede conducir a la eliminación de algunos cuya información ya esté presente en otro atributo.

Preprocesamiento de datos

Como en el problema de clasificación, procedemos a procesar los datos para reducir asimetrías, escalarlos y centrarlos. De nuevo, como trabajamos con un número pequeño de atributos, no aplicaremos PCA para la reducción de atributos.

A continuación, vemos de nuevo la distribución de atributos 2 a 2 una vez preprocesados los datos. Comprobamos, como ya se comentó en la clasificación, que, aunque hayan cambiado las escalas y las posiciones de los datos en las gráficas (basta ver los valores que muestran los ejes), las formas en las que se distribuyen los datos en las gráficas siguen siendo las mismas:

```
## The following objects are masked from ozone.df:
##
##     doy, dpg, humidity, ibh, ibt, ozone, temp, vh, vis, wind
```



Conjuntos de validación, training y test usados.

A continuación pasaremos a explorar los distintos modelos sobre los que resolver el problema de regresión para nuestro conjunto de datos. Al igual que en regresión, el procedimiento de validación que usaremos consistirá en tomar múltiples veces distintos conjuntos de entrenamiento sobre nuestro conjunto de datos (una vez transformados), con los que aprenderemos el modelo. Usaremos el resto del conjunto como test para evaluar cómo de bien predice el modelo aprendido con nuevos datos. Las proporciones utilizadas serán de nuevo del 70 % de los datos para train, y el 30 % para test.

Selección de clases de funciones a usar

Procedemos ahora a la selección de modelos de regresión. El modelo seleccionado para este caso es el de regresión lineal clásico, que podemos utilizar en `R` mediante la función `lm`.

Una vez definido el modelo, procedemos a su ajuste y al análisis de los errores obtenidos:

```
##           Ein     Eout
## [1,] 19.19426 21.12787
```

Regularización.

Al igual que en el problema de clasificación, obtendremos dos valores de lambda y compararemos los errores producidos con cada uno de estos valores y con el modelo anterior, sin aplicar regularización.

Los errores producidos han sido:

```
## Etest con lambda min:  20.72697
```

```
## Etest con lambda 1se:  22.50946
```

```
## Etest con el modelo original:  20.78467
```

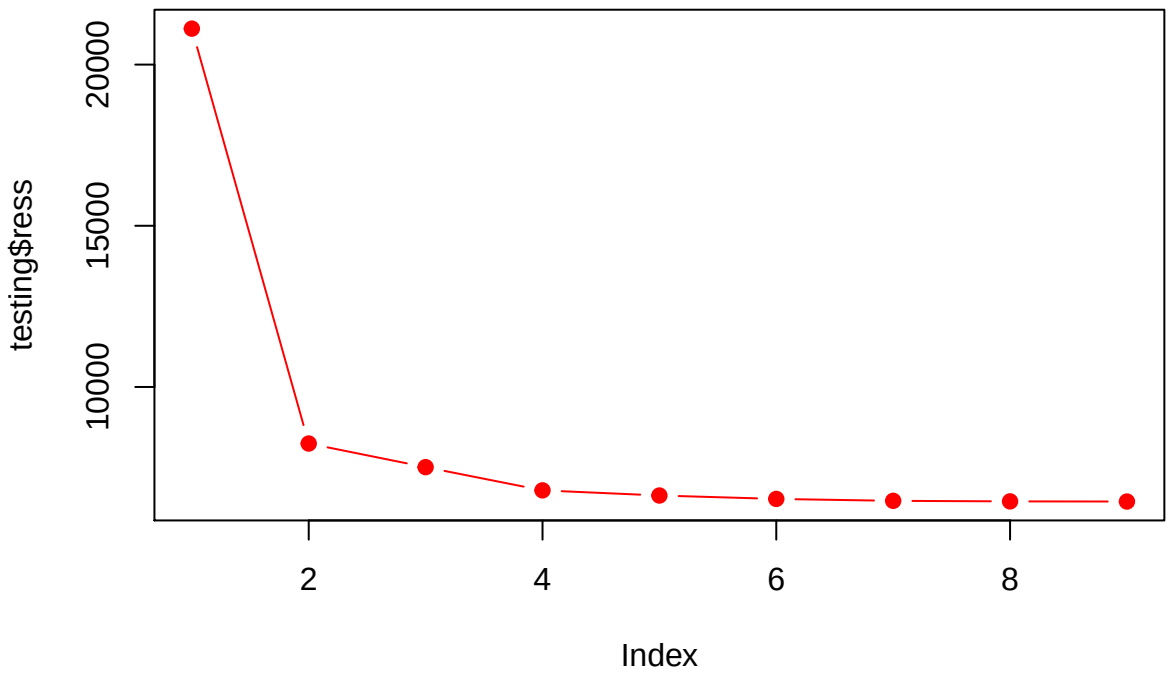
A diferencia de en el problema de clasificación, hemos obtenido que el menor error se consigue con *lambda.min*, aunque la diferencia con el modelo sin regularización es insignificante, y por tanto la regularización no es necesaria.

Optimización del número de atributos para el modelo seleccionado

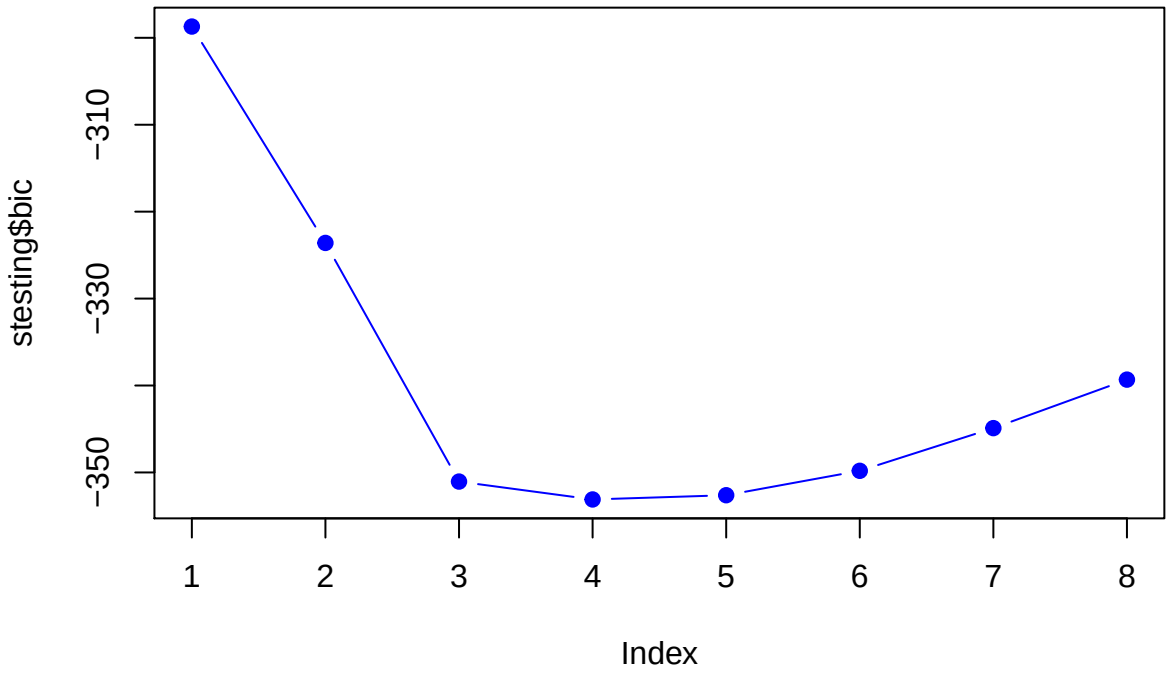
Al igual que en la clasificación, vamos a elegir el número de atributos que consideramos para nuestro modelo. Análogamente, realizaremos una búsqueda exhaustiva de las mejores combinaciones de atributos. Obtenemos que el esquema resultante es el siguiente:

```
##          vh  wind humidity temp ibh dpg ibt vis doy
## 1  ( 1 ) " " " "  " "      "*"  " " " " " " " " "
## 2  ( 1 ) " " " "  " "      "*"  "*" " " " " " " " "
## 3  ( 1 ) " " " "  "*"      "*"  "*" " " " " " " " "
## 4  ( 1 ) " " " "  "*"      "*"  " " " " " " "*" " " "*"
## 5  ( 1 ) " " " "  "*"      "*"  " " " " " " "*" "*" "*"
## 6  ( 1 ) "*" " "  "*"      "*"  " " " " " " "*" "*" "*"
## 7  ( 1 ) "*" " "  "*"      "*"  "*" " " " " "*" "*" "*"
## 8  ( 1 ) "*" " "  "*"      "*"  "*" "*" "*" "*" "*" "*"
```

Dibujamos también la gráfica descendente de errores *ress*:



De forma análoga al ejercicio anterior, dibujamos la gráfica de *BIC* para determinar qué número de atributos optimiza el error junto con el número de atributos obtenido.



En este caso, aunque no es tan evidente como en el ejercicio de clasificación, vemos que el mínimo de la función *BIC* se alcanza en cuatro atributos, luego nos quedamos con la mejor combinación de cuatro atributos. Esta es, la que observamos en el esquema del *regsubsets*: humidity, temp, ibt, doy.

Transformación de atributos

Como hicimos en clasificación, procedemos ahora a buscar transformaciones polinómicas que permitan un mejor ajuste de los datos, recurriendo para ello al mismo algoritmo greedy utilizado en clasificación.

```
## Attr =  1 , Exp =  1 Ein =  19.87888 , Eout =  21.10823
## Attr =  1 , Exp =  2 Ein =  21.36381 , Eout =  22.49587
## Attr =  1 , Exp =  3 Ein =  20.05716 , Eout =  21.07052
## Attr =  1 , Exp =  4 Ein =  21.41946 , Eout =  22.77382
## Attr =  2 , Exp =  1 Ein =  19.80816 , Eout =  21.63701
## Attr =  2 , Exp =  2 Ein =  21.04126 , Eout =  22.70644
## Attr =  2 , Exp =  3 Ein =  22.40218 , Eout =  23.64018
## Attr =  2 , Exp =  4 Ein =  21.66081 , Eout =  23.33565
## Attr =  3 , Exp =  1 Ein =  20.06859 , Eout =  21.03871
## Attr =  3 , Exp =  2 Ein =  21.15955 , Eout =  22.10031
## Attr =  3 , Exp =  3 Ein =  21.89371 , Eout =  22.61509
## Attr =  3 , Exp =  4 Ein =  21.26019 , Eout =  22.07568
## Attr =  4 , Exp =  1 Ein =  20.0846 , Eout =  21.06911
## Attr =  4 , Exp =  2 Ein =  19.18092 , Eout =  20.33289
```

```
## Attr =  4 , Exp =  3 Ein =  20.7418 , Eout =  21.66409
## Attr =  4 , Exp =  4 Ein =  19.88932 , Eout =  20.61432
```

```
## Vector de exponentes:  3 1 1 2
```

```
## Eout estimado:  20.33289
```

En este caso vemos que sí hay transformaciones polinómicas que mejoran el ajuste, luego esta transformacion polinómica será la elegida para el modelo final.

A modo de ampliación, aunque ya habíamos reducido nuestro conjunto de datos a cuatro atributos, vamos a probar las trasnformaciones no lineales sobre todos los atributos.

```
## Attr =  1 , Exp =  1 Ein =  19.14471 , Eout =  21.41
## Attr =  1 , Exp =  2 Ein =  18.55248 , Eout =  20.06719
## Attr =  1 , Exp =  3 Ein =  19.17528 , Eout =  20.4578
## Attr =  1 , Exp =  4 Ein =  18.94393 , Eout =  21.05477
## Attr =  2 , Exp =  1 Ein =  18.33188 , Eout =  20.56382
## Attr =  2 , Exp =  2 Ein =  18.50706 , Eout =  20.64506
## Attr =  2 , Exp =  3 Ein =  18.37103 , Eout =  21.98909
## Attr =  2 , Exp =  4 Ein =  18.57959 , Eout =  30.19729
## Attr =  3 , Exp =  1 Ein =  18.56776 , Eout =  20.02829
## Attr =  3 , Exp =  2 Ein =  19.04625 , Eout =  21.35706
## Attr =  3 , Exp =  3 Ein =  18.71656 , Eout =  20.47936
## Attr =  3 , Exp =  4 Ein =  19.37991 , Eout =  20.67961
## Attr =  4 , Exp =  1 Ein =  18.63648 , Eout =  19.81035
## Attr =  4 , Exp =  2 Ein =  19.4225 , Eout =  21.65299
## Attr =  4 , Exp =  3 Ein =  19.70035 , Eout =  21.74068
## Attr =  4 , Exp =  4 Ein =  20.03524 , Eout =  21.74907
## Attr =  5 , Exp =  1 Ein =  18.60192 , Eout =  19.94308
## Attr =  5 , Exp =  2 Ein =  18.46437 , Eout =  20.07576
## Attr =  5 , Exp =  3 Ein =  18.34155 , Eout =  20.49264
## Attr =  5 , Exp =  4 Ein =  18.46067 , Eout =  20.07442
## Attr =  6 , Exp =  1 Ein =  18.51967 , Eout =  20.15759
## Attr =  6 , Exp =  2 Ein =  17.3368 , Eout =  19.65353
## Attr =  6 , Exp =  3 Ein =  18.42193 , Eout =  20.34311
## Attr =  6 , Exp =  4 Ein =  18.1996 , Eout =  19.28127
## Attr =  7 , Exp =  1 Ein =  17.90641 , Eout =  20.07319
## Attr =  7 , Exp =  2 Ein =  18.07566 , Eout =  19.1697
## Attr =  7 , Exp =  3 Ein =  18.47668 , Eout =  20.44054
## Attr =  7 , Exp =  4 Ein =  17.94332 , Eout =  19.9358
## Attr =  8 , Exp =  1 Ein =  17.91996 , Eout =  19.5633
## Attr =  8 , Exp =  2 Ein =  18.42752 , Eout =  19.80616
## Attr =  8 , Exp =  3 Ein =  18.39251 , Eout =  19.85114
## Attr =  8 , Exp =  4 Ein =  18.66299 , Eout =  19.29149
## Attr =  9 , Exp =  1 Ein =  18.24714 , Eout =  20.27092
## Attr =  9 , Exp =  2 Ein =  17.85082 , Eout =  19.62415
## Attr =  9 , Exp =  3 Ein =  18.82199 , Eout =  19.81203
## Attr =  9 , Exp =  4 Ein =  18.34249 , Eout =  19.51563
```

```
## Vector de exponentes:  2 1 1 1 1 4 2 4 4
```

```
## Eout estimado:  19.51563
```

Como podemos observar, en este caso hemos obtenido un error menor al obtenido anteriormente, aunque esto se ha hecho a expensas de utilizar más atributos, lo que conlleva una mayor carga de cómputo. Aunque en este ejercicio no es un problema dado la reducida dimensión del conjunto de datos a utilizar, es importante tener un balance entre el error producido y el número de atributos considerado.

Conclusiones

Por los resultados que hemos ido obteniendo anteriormente, el modelo final escogido sería aplicar el método *lm* a la base de dats con los cuatro atributos: humidity, temp, ibt, doy. El error estimado sería del 20.05149%.

En cuanto a las transformaciones no lineales obtenidas sobre todos los atributos, podemos ver que se corresponden con la relación entre ozone y el resto de atributos que podemos observar en la tabla de la sección de Preprocesamiento de datos. Observamos que los atributos vh, dpg, ibh y doy tienen un comportamiento cuadrático. El comportamiento de doy se confirma cuando probamos transformaciones no lineales sobre los cuatro atributos escogidos para el modelo final.