Capítulo 1

Diagnosticos del Modelo Lineal

Una vez que hacemos un ajuste lineal debemos verificar los supuestos que impuso toda la teoría que contruyó los estimadores así como las distintas pruebas de hipósteis de significancia.

El principal supuesto y sobre el cual reace toda la teoria es suponer que dentro del modelo se tiene que:

$$\underline{\varepsilon} \sim N_n\left(\mathbf{X}\underline{\beta}, \sigma^2\mathbb{I}\right)$$

De lo anterior, surge la necesidad de crear pruebas y herramientas estadísticas que nos ayuden a *defender* el modelo que construimos.

1.1. Validación de supuestos

Los principales supuestos que debemos verificar cuando ajustamos un modelo es:

- Linealidad entre la variable respuesta y las variables explicativas
- Independencia de los residuales (ACF, DurbinWatson, Rachas)
- Varianza Constante de los residuales (Homocedasticidad, Barttlet, Levene)
- Normalidad de los residuales (Histograma y QQPlot)
- Multicolinealidad (Rango completo de la matriz diseño, VIF)
- *Outliers* y observaciones influyentes

1.1.1. Residuales vs ε_i

Recordemos que nuestro modelo general es de la forma:

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \varepsilon_i$$

Luego entonces:

$$\varepsilon_i = y_i - \beta_0 - \beta_1 x_{ii} - \ldots + \beta_k x_{ik}$$

Sin embargo $(\beta_0, \ldots, \beta_p)$ son desconocidos luego entonces ε_i no son observables al momento de hacer los análisis. Tenemos el problema de que algunos de los supuestos del modelo recaen en suponer que hay Normalidad y Homocedasticidad en las ε_i. Para solucionar esto se suele utilizar a los residuales como una aproximación a la realización de las variables ε_i y por tanto el análisis recaé sobre el vector de residuales definido como:

$$\underline{e} = \underline{Y} - \underline{\hat{Y}} = \underline{Y} - \mathbf{X}\underline{\hat{\beta}} = (\mathbf{I} - \mathbf{H})\,\underline{Y}$$

Ahora bien, recordando que $\underline{Y} = \mathbf{X}\underline{\beta} + \underline{\varepsilon}$ entonces:

$$\underline{e} = (\mathbf{I} - \mathbf{H})\,\underline{Y} = (\mathbf{I} - \mathbf{H})\left(\mathbf{X}\underline{\beta} + \underline{\varepsilon}\right) = (\mathbf{I} - \mathbf{H})\,\underline{\varepsilon}$$

Por lo tanto

$$\mathbf{var}\,(\underline{e}) = \mathbf{var}\,((\mathbf{I} - \mathbf{H})\,\underline{\varepsilon}) = \sigma^2\,(\mathbf{I} - \mathbf{H})$$

1.1.2. Verificación del Supuesto de Linealidad

Recordemos que el modelo lineal que ajustamos es de la forma:

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \varepsilon_i$$

Si $k = 1$ (Modelo lineal simple) es relativamente fácil verificar linealidad, basta con realizar una gráfica de los puntos (y_i, x_{i1}) para verificar si en realidad existe una recta asociada a la relación.

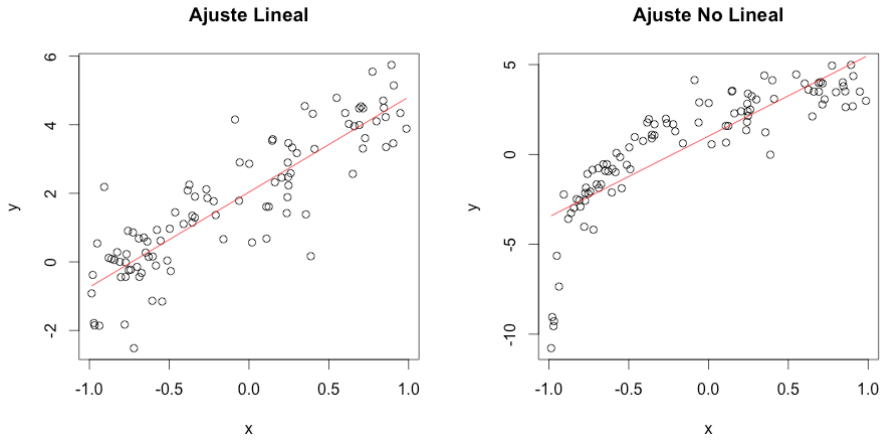


Figura 1.1: Verificación de la linealidad

Cuando tenemos 2 o mas variables explicativas la verificación de linealidad se complica debido a que se requiere graficar en dimensiones mayores. Una solución inicial al problema es graficar cada una de las variables explicativas contra la variable respuesta y verificar ahi la linealidad, sin embargo dichos graficos pueden llevarnos a conclusiones erróneas cuando los coeficientes tienen magnitudes distintas. En la siguiente figura se muestran las gráficas de las covariables vs la variable respuesta de un modelo **lineal**

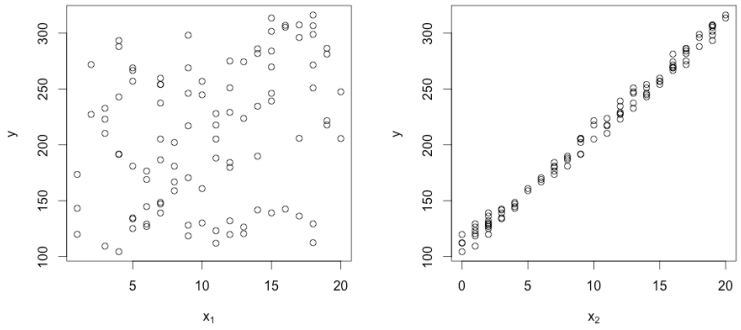


Figura 1.2: Verificación de la linealidad

Observe que en gráfico anterior, a pesar de que el modelo si es lineal, **no** se logra identificar la lienalidad de la variable x_1.

Para solucionar esto recurriremos al **análisis de residuales parciales**.
Supongamos que queremos ver si existe una relación lineal entre la variables X_q y Y, entonces lo que haces es ajustar el modelo **eliminando** a la variable X_q del modelo, luego encontramos los residuales de dicho modelo (denotamos por $e_i^{(q)}$) y luego graficamos esos residuales contra los valores de la variables X_q, es decir visualizamos los puntos $\left(x_{iq}, e_i^{(q)}\right)$. Si la relación es lineal entre X_q y Y se espera que esta gráfica de residuales parciales presente una relación lineal, en caso contrario, es una indicación de que esa variable posiblemente necesite ser transformada para ayudar al ajuste.

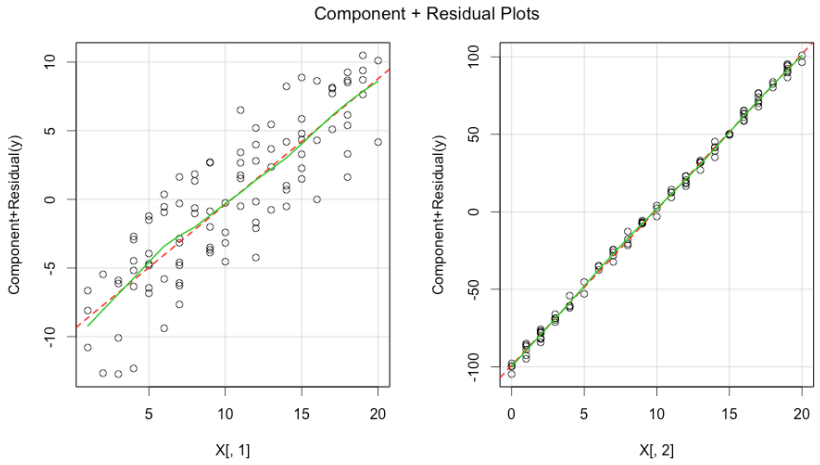


Figura 1.3: Residuales Parciales

Observe en el gráfico anterior que ahora si se observa una relación lineal entre todas las variables explicativas por lo que se valida la linealidad del modelo.

1.1.3. Verificación del Supuesto de Independencia

Generalemente para verificar que tenemos independencia de los errores se suele hacer una gráfica de los residuales contra el orden de obtención de los datos. Luego en el gráfico generado se busca algún tipo de patrón que evidencíe la dependencia de los errores conforme se van realizando las mediciones.

Sin embargo a veces no es fácil detectar un patrón por lo que se pueden aplicar pruebas estadísticas para detectar patrones en los datos, las pruebas mas usuales son las que útilizan la función de autocorrelación **(ACF)** y la prueba no parámetrica de **Rachas** que va midiendo el cambio de los signos de los errores.

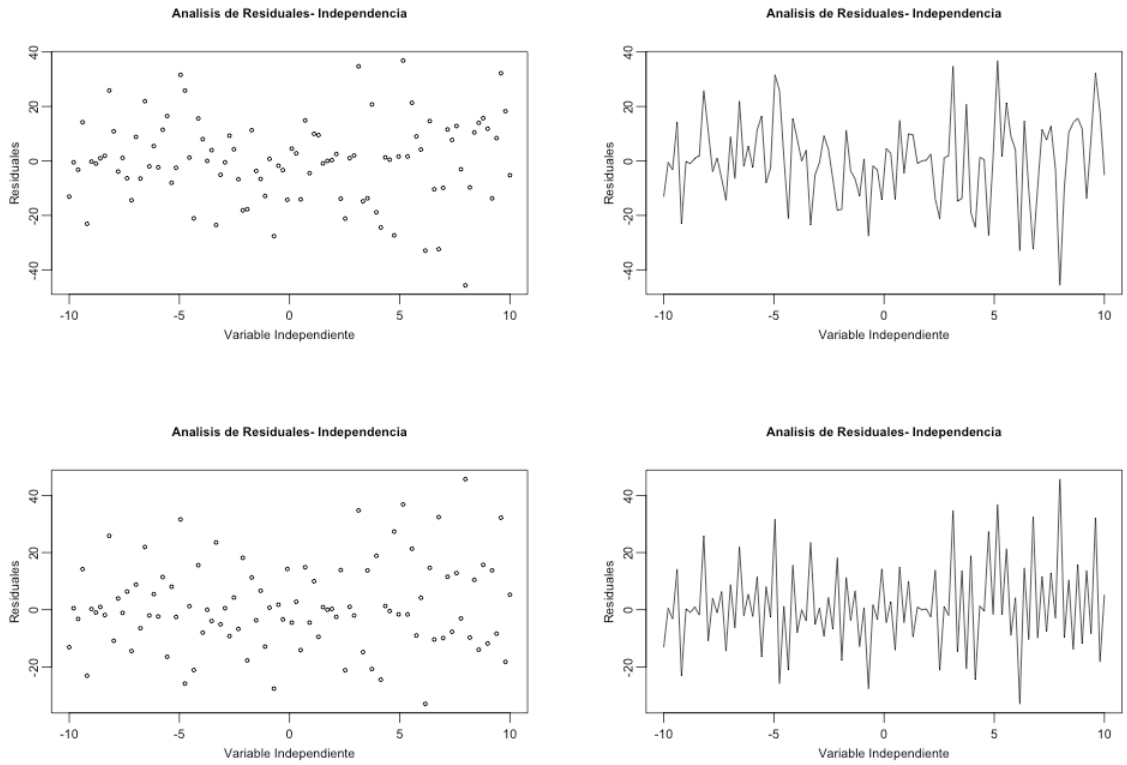


Figura 1.4: Independencia

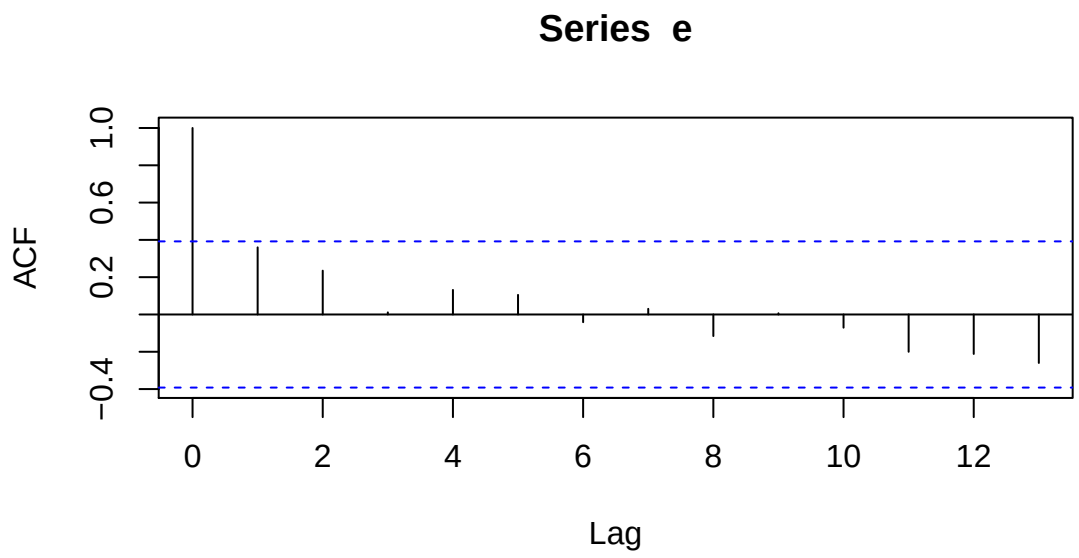
- **Función de Autocorrelación (ACF).**
  En estadística, la autocorrelación de una serie de datos discreta de un proceso X_t no es más que el coeficiente de correlación de dicho proceso con una versión desplazada de la propia serie. La forma en como se calcula es la siguiente: suponiendo que tenemos a la serie de datos $(e_1, e_2, \ldots, e_n)$ entonces la función de autocorrelación con rezago k se obtiene como:

$$\rho_k = \frac{\sum_{i=1}^{n-k} (e_i - \bar{e})(e_{i-k} - \bar{e})}{\sum_{i=1}^{n} (e_i - \bar{e})^2}$$

  Si ρ_k se aleja del valor 0 hay evidencia de que cada k observaciones hay un patrón de los residuales pues dichas observaciones están muy correlacionadas lo que implicaría dependencia de los residuales. La idea entonces es contrastar la hipótesis $\rho_k = 0$ vs $\rho_k \neq 0$, afortunadamente R tiene programada la prueba en la función *acf*.
  Ejemplo:

```
# Ejemplo Regresion Multiple Lectura de Datos de tiempo de entrega.
tiempos <- read.csv("/Users/act_antonio/Documents/Tiempos.csv")
# Asigmanos variable respuesta a Y
y = tiempos[, 1]
```

```
# Creamos matriz de dise<U+00F1>o
X = cbind(rep(1, 25), tiempos[, 2], tiempos[, 3])
# Definimos tama<U+00F1>os
n = 25
p = 2
# Corremos modelo lineal, (Observe que eliminamos la columnas de 1's en X)
model = lm(y ~ X[, 2:3])
# Extraigo residuales
e = model$res
# Prueba de Independencia
acf(e)
```



En esta gráfica esperamos que todas todas las correlaciones (excepto la primera que siempre toma el valor de 1) estén dentro de la banda de confianza que crea el paquete estadístico. En caso de que alguna observación se salga de la banda se considera que se encontró una correlación estadísticamente significativa y por tanto se dice que hay un patrón en los residuales lo que se traduce en que no hay independencia.

Un punto en contra de esta prueba es que requiere del supuesto de Normalidad para poder concluir independencia, luego entonces para que la prueba sea válida se require antes justificar la normalidad de los residuales.

- **La prueba de Rachas**

  Otra forma que tenemos de verificar que los residuales son independientes es probar la aleatoreidad con la que van cambiando de signo los errores, si los residuales son independientes

se esperaría que el cambio de signo del residual conforme se va obteniendo la muestra sea aleatorio.

Definición 1.1.1 (Racha) *Se considera una Racha de tamaño k a la secuencia de k valores consecutivos de un mismo signo siempre y cuando estos sean precedidos y seguidos por valores con signo opuesto a la de la Racha.*

$$\underbrace{+,+}, \underbrace{-}, \underbrace{+}, \underbrace{-}$$

La idea de la prueba es contar el número de rachas en la muestra, luego un número reducido o grande de rachas es indicio de que las observaciones no se han obtenido de forma aleatoria. Si la muestra es grande y la hipótesis de aleatoriedad es cierta la distribución muestral del número de rachas R, puede aproximarse mediante una distribución Normal de parámetros:

$$\mu_R = \frac{2n_1n_2}{n} \quad \sigma_R^2 = \frac{2n_1n_2(2n_1n_2 - n)}{n^2(n-1)}$$

La prueba de Rachas se encuentra programada de R y se ubica dentro de la librería *tseries* bajo el nombre *runs.test*, esta función recibe un vector con valores binarios indicando el signo del residual.
Ejemplo:

```
library(tseries)
runs.test(as.factor(e > 0))
```

```
##
##  Runs Test
##
## data:  as.factor(e > 0)
## Standard Normal = -1.015, p-value = 0.3101
## alternative hypothesis: two.sided
```

En estas pruebas se espera tener p-values *grandes* que dan indicio de que la prueba no debe de ser rechazada lo que justifica la aleatoriedad con la que van saliendo los residuales y por tanto justifica la independencia de los mismos.

- **Prueba de Durbin Watson**
  Otra prueba muy utilizada para la identificación de independencia (no correlación) es utilizar

la prueba de *Durbin Watson*. Esta prueba pretende ver si los valores presentan algún tipo de dependencia en cuanto al orden de obtención. La prueba se plantea como sigue: Supongamos que los errores ε_i siguen un modelo AR(1) entonces:

$$\varepsilon_i = \rho\varepsilon_{i-1} + \delta_i \qquad \delta_i \sim N(0, \sigma^2)$$

Luego entonces planteamos:

$$H_0 : \rho = 0 \qquad vs \qquad H_1 : \rho \neq 0$$

Estadístico de prueba:

$$d = \frac{\sum_{i=2}^{n}(e_i + e_{i-1})^2}{\sum_{i=1}^{n} e_i^2}$$

El comando utilizado en R es *durbinWatsonTest(model)* y esperamos ver un p-value alto para no rechazar. (Ojo: Estas prueban suponen Normalidad).
Ejemplo:

```
library(car)
durbinWatsonTest(model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1          0.3599         1.168   0.012
##  Alternative hypothesis: rho != 0
```

1.1.4. Verificación del Supuesto de Homocedasticidad

Generalmente para verificar que tenemos varianza constante se llevan a cabo pruebas visuales verificando el comportamiento de los residuales conforme se van observando, o bien graficar contra $\hat{y}_i$.
Dado que ahora tenemos mas variables explicativas, también se suele graficar los residuales contra cada una de las variables explicativas y verificar le homocedasticidad ahi.

Afortunadamente existen pruebas formales para verificar la homocedasticidad (Ej. Barttlet, Levene) pero requieren tener definido grupos lo cual puede quitarle credibilidad a la prueba. (Ojo: Las pruebas suponen Normalidad para que los resultados sean confiables).
La hipotesis nula de estas pruebas es que hay homocedasticidad (igualdad de varianzas) por lo que se contrasta lo siguiente:

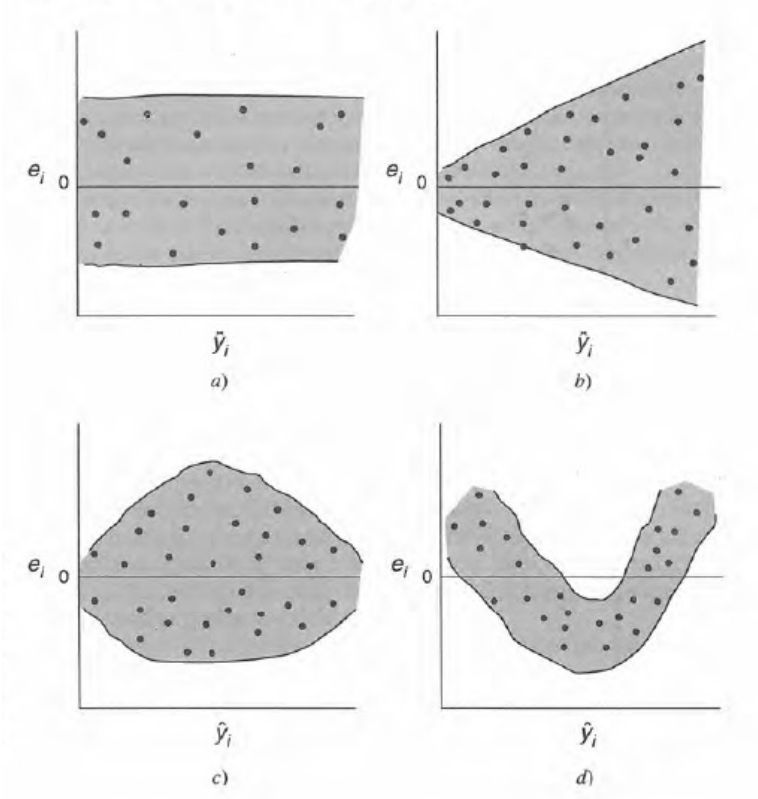


Figura 1.5: Homocedasticidad

$$H_0 : Los - datos - son - homocedasticos \quad vs \quad H_1 : No - hay - homocedasticidad$$

En este sentido lo que se desea es no rechazar H_0 y por tanto se esperan ver p-values grandes Si se fija $\alpha = 0.05$, ver p-values mas grandes que 0.05 hacen no rechazar la homocedasticidad

- **Prueba Barttlet**
  Esta prueba requieren tener definido grupos en el vector de los residuales, por lo general esta prueba es utilizada en diseño de experimentos pero podemos adaptarla al caso de regresión definiendo grupos arbitrarios, el problema que tiene esto es que deja que el analista determine el número de grupos así como el método para formarlos, lo cual puede no ser objetivo y llegar a cometer errores.
  En el paquete R podemos correr esta prueba utilizando el comanado *bartlett.test(e,g)* donde en e se guardan los residuales y en g se guardan los grupos a donde pertenece cada residual, el vecto g entonces debe ser previamente definido por el analista.
  Ejemplo:

```
library(car)
g = c(rep(1, 8), rep(2, 8), rep(3, 9))
bartlett.test(e, g)

##
##  Bartlett test of homogeneity of variances
##
## data:  e and g
## Bartletts K-squared = 0.397, df = 2, p-value = 0.82
```

Observe que como en el vector g se forman tres grupos, el primero y el segundo de 8 observaciones y el último de 9 observaciones. La formacioń de los grupos es en función al orden de observación.

- **Prueba Levene**
  Al igual que la prueba de Barttlet, Levene requiere la definición de grupos para el correcto funcionamiento de la prueba. La idea que hay de tras de la prueba es utilizar el ANOVA para probar iguadad de las varianzas en cada grupo. En el paquete R podemos correr esta prueba utilizando el comando *levene.test(e,g)*
  Ejemplo:

```
library(car)
g = c(rep(1, 8), rep(2, 8), rep(3, 9))
levene.test(e, g)

## Warning: 'levene.testís deprecated.
## Use 'leveneTestínstead.
## See help("Deprecated") and help("car-deprecated").
## Warning: g coerced to factor.

## Levenes Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  2    0.78   0.47
##       22
```

1.1.5. Verificación del Supuesto de Normalidad

Parte fundamental de la validación del modelo es verificar la normalidad de los residuales, para ello contamos con pruebas de bondad de ajuste, sin embargo una primera prueba se hace llevando

a cabo un histograma de los residuales o de QQ plot para verificar que siguen una distribución parecida a la normal.

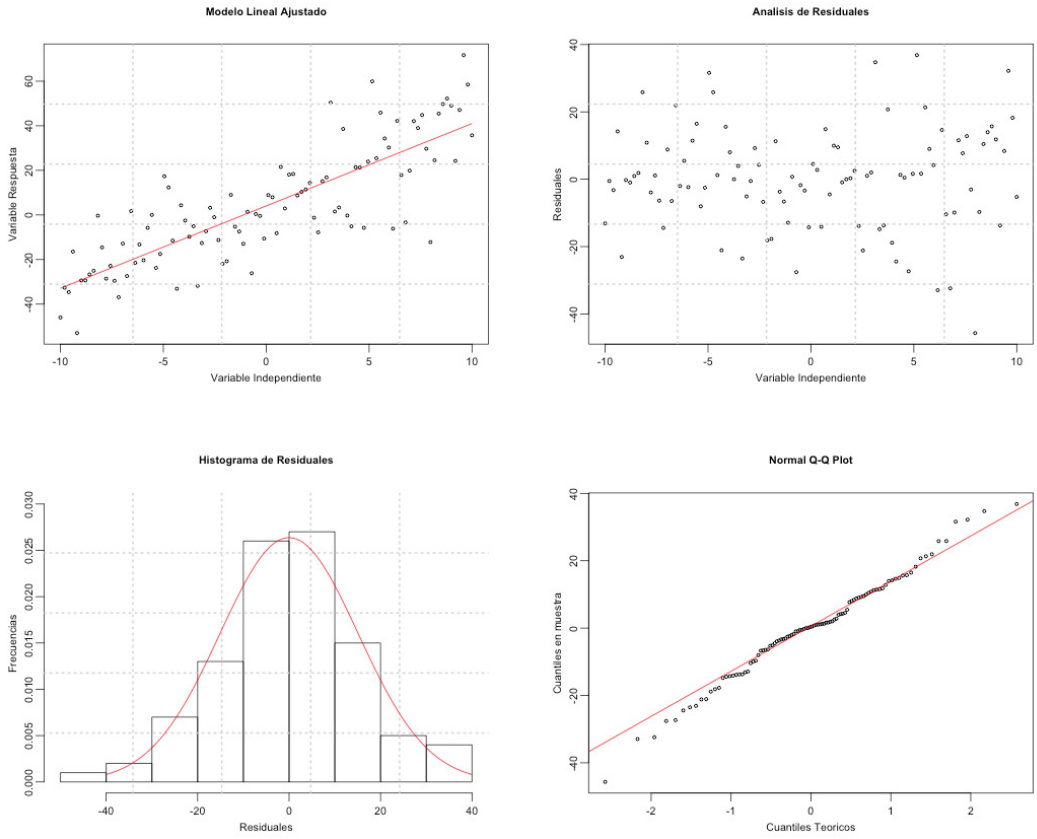


Figura 1.6: Prueba de Normalidad

Existen muchas pruebas de bondad de ajuste para verificar la normalidad, las mas usadas son:

- Shapiro-Wilk Normality Test (shapiro.test(x))
- Anderson-Darling test for normality (ad.test(x))
- Liliiefors for normality (lillie.test)

La idea de todas las puebas es contratastar la hipótesis:

$$H_0 : e_i \sim N(\mu, \sigma^2) \quad vs \quad H_1 : e_i \nsim N(\mu, \sigma^2)$$

Ejemplo:

```
library(nortest)
ad.test(e)

##
##  Anderson-Darling normality test
##
## data:  e
## A = 0.5959, p-value = 0.1091

lillie.test(e)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  e
## D = 0.1776, p-value = 0.04073

cvm.test(e)

##
##  Cramer-von Mises normality test
##
## data:  e
## W = 0.1095, p-value = 0.07776
```

Sin embargo, teóricamente las pruebas de bondad de ajuste no pueden ser aplicadas pues existe correlación entre los residuales y por tanto se esta violando el supuesto de la prueba de bondad de ajuste de tener variables independientes. Es por ello que estas pruebas deben de utilizarse solo como una medida de que tan normales son los residuales y no como regla de decisión basada en un nivel de significancia.

1.1.6. Multicolinearidad

Dentro de los supuesto del modelo lineal se requirió que la matriz de diseño $\mathbf{X}$ tuviera rango completo, esto con el fin de garantizar la invertibilidad de la matriz $\mathbf{X}^t\mathbf{X}$ y obtener una solución única a las ecuaciones normales $\left(\underline{\hat{\beta}} = \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\underline{Y}\right)$. El hecho de pedir rango completo a la matriz $\mathbf{X}$ nos obliga a verificar que no existe columnas en $\mathbf{X}$ que sean combinación lineal de otras. En la práctica esto casi no ocurre sin embargo lo que si suele ocurrir es que existan columnas que *casi* son combinaciones lineales de otras columnas. Teóricamente al no haber un igualdad exacta se garantiza

la existencia de la matriz inversa. Sin embargo numéricamente esto ocasiona problemas.

$$\mathbf{X} = (\underline{X}_0, \underline{X}_1, \ldots, \underline{X}_k)$$

$$\underline{X}_q \approx \alpha_0 \underline{X}_0 + \ldots + \alpha_{q-1}\underline{X}_{q-1} + \alpha_{q+1}\underline{X}_{q+1} + \ldots + \alpha_k \underline{X}_k$$

Verificar la multicolineridad no es fácil a simple vista sin embargo existen formas de detectarla.

- Una forma muy simple para detectar multicolinearidad es inspeccionar la matriz de correlaciones entre las variables explicativas. (`cor(X) , pairs(X)`)
- Hacer regresiones lineales de X_q con el resto de las variables y obtener el coeficiente de determinación de dicha regresión R_q^2, un coeficiente de deteminación cercana a 1 nos dice que hay un ajuste muy bueno por lo que se puede decir que:

  $$\underline{X}_q \approx \alpha_0 \underline{X}_0 + \ldots + \alpha_{q-1}\underline{X}_{q-1} + \alpha_{q+1}\underline{X}_{q+1} + \ldots + \alpha_k \underline{X}_k$$

  Una forma de obtener indices es calcular lo que se denomina el VIF_q (Factor de Inflación de la varianza) definido como:

  $$VIF_q = \frac{1}{1 - R_q^2} \quad q \in \{0, 1, \ldots, k\}$$

  Luego entonces si $R_q^2 \approx 1$ eso se traduce en un VIF_q grande, en la práctica se suele tomar como punto de corte 5 ó 10, es decir obtener un $VIF_q > 10$ nos habla de que la variable X_q puede ser expresada aproximadamente como combinación lineal de las restantes y por tanto podemos tener problemas numéricos en las estimaciones.
  En el paquete R los VIF pueden ser calculados mediante la función vif.

  Ejemplo:

```
DATA = as.data.frame(cbind(y, X[, 2:3]))
names(DATA)[2] = "cajas"
names(DATA)[3] = "distancias"
model = lm(y ~ ., data = DATA)
library(car)
vif(model)

##      cajas distancias
##      3.118      3.118
```

Note que para correr correctamente la función de R requerimos correr el modelo lineal utilizando DATAFRAMES.

- Otra forma de detectar la multicolinearidad es con el índice de condición de una matriz $\left(\mathbf{X}^t\mathbf{X}\right)$ el cual se definie como:

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}} \qquad kappa\left(t(X)\,\%*\%X\right)$$

  Generalmente un indice de condición menor a 100 no tiene problemas de multicolinealidad, si está entre 100 y 1000 implica una moderada multicolinearidad y si excede a 1000 entonces hay problemas importantes y podemos tener problemas númericos. La forma como se calcula en R es mediante la función *kappa*

- También se estila calcular el determinante de la matriz $\left(\mathbf{X}^t\mathbf{X}\right)$, un determinante cercano a 0 nos habla de problemas numéricos al momento de invertir.

1.2. Outliers y Observaciones influyentes

Hay dos propósitos principales para tratar de identificar las observaciones que no pertenecen al modelo

- Proteger la integridad del modelo de los efectos de los puntos que no pertenecen a éste. (Observaciones que no pertenecen al modelo) Frecuentemente exhiben residuales grandes.

- Identificar las deficiencias del modelo . Posiblemente se necesite incluir nuevas variables o se requiere llevar a cabo transformaciones.

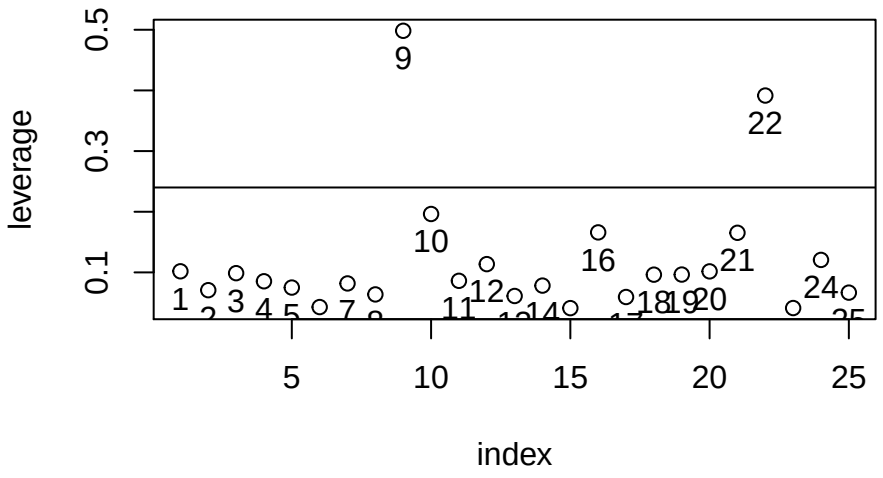
El primer paso para identificar observaciones atípicas es tener una idea de la región en donde estamos muestreando, la idea es encontrar una métrica que nos ayude a decidir si una observación de las variables independientes está muy alejada de la región de muestreo. En general una observación que se encuentra lejos de donde esta la masa de puntos ocasiona que el modelo cambie de manera significativa (Aunque no necesariamente).

Para detectar los puntos de muestreo que están alejados se utiliza la matriz sombrero, $\mathbf{H} = \mathbf{X}\left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t$. A los elementos de la diagonal de $\mathbf{H}$ (h_{ii}) se les conoce como el *leverage* o la influencia de cada observación. De este modo un *leverage* grande nos habla de una observación alejada de la masa de los puntos de muestreo.
Dado que sabemos que:

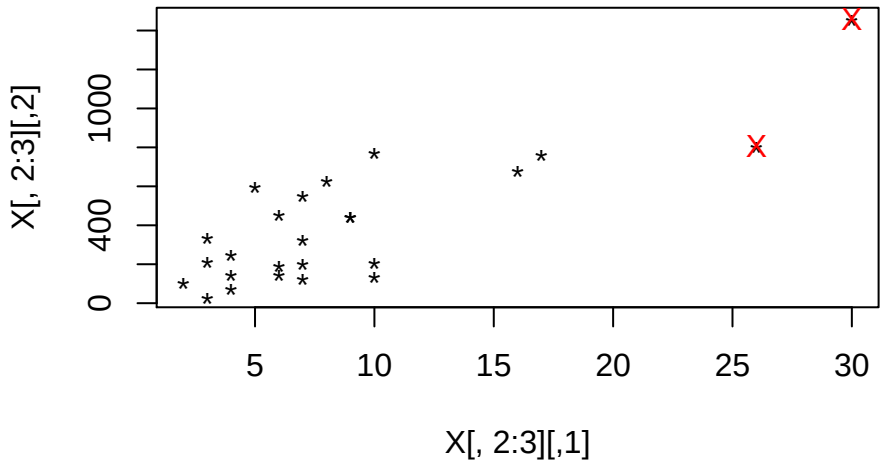
$$tr(\mathbf{H}) = \sum_{i=1}^{n} h_{ii} = p$$

Se concluye que en promedio los *leverages* toman el valor de p/n. Existe una regla muy utilizada que dice que aquellos h_{ii} que sean mayores a dos veces al promedio son puntos alejados y por tanto deben de ser analizados. ($h_{ii} > 2p/n$)
Ejemplo:

```
# Leverage
p = 2
n = length(X[, 1])
# La funcion hat obtiene la diagonal de matriz sombrero
h = hat(X)
plot(h, xlab = "index", ylab = "leverage")
abline(h = 2 * (p + 1)/n)
text(h, pos = 1)
```




```
# Esto nos indica que las observaciones 9 y 22 estan relativamente alejados
# de la zona de muestra
plot(X[, 2:3], pch = "*", col = 1)
points(X[c(9, 22), 2:3], pch = "X", col = 2)
```




```
# Las observaciones en Rojo estan alejados y por tanto son candidatos a ser
# obseravaciones influyentes
```

1.2.1. Outliers

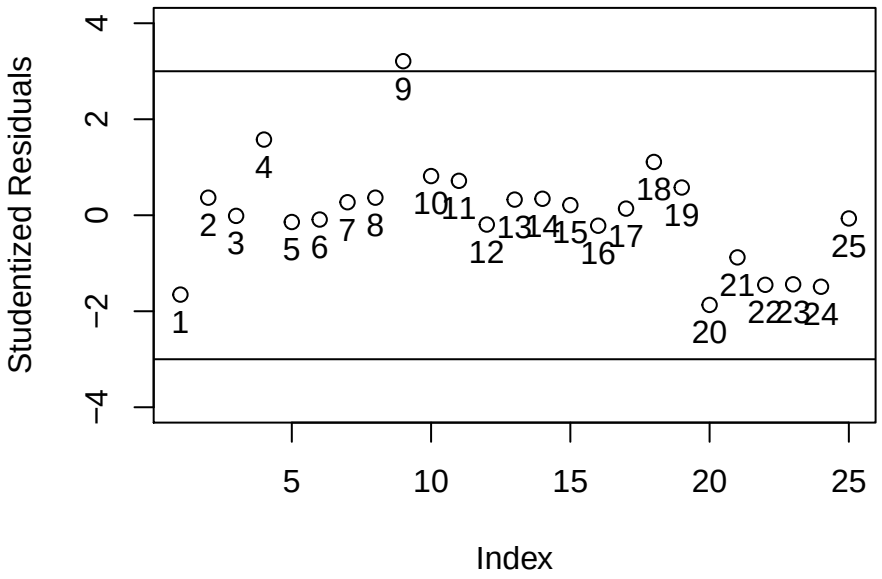
Definición 1.2.1 (Outlier) *Es un punto que no ajusta en el modelo.*

Surge la pregunta: ¿Como detectar outliers?
Un primer intento para encontrar este tipo de observaciones consiste en *studentizar* a los residuales:

$$r_i = \frac{e_i}{\hat{\sigma}\sqrt{1 - h_{ii}}}$$

Luego se espera que estos residuales r_i se encuentren en una banda entre -3 y 3, aquellos fuera de estas bandas serán candidatos a ser analizados como outliers del modelo.
Ejemplo:

```
res = model$res
s.2 = anova(model)$"Mean Sq"[3]
resstud = res/sqrt(s.2 * (1 - h))
plot(resstud, xlab = "Index", ylab = "Studentized Residuals", ylim = c(-4, 4))
abline(h = c(-3, 3))
text(resstud, pos = 1)
```



En el ejemplo anterior la observación 9 será considerado como Outlier

El método anterior tiene el problema de que posiblemente el modelo ajustado este siendo influenciado por el dato outlier y por tanto no sea detectado en los residuales, para solucionar esto se estila utilizar lo que se denomina el residual de *jackknife* (Residual de validación cruzada)

$$t_i = \frac{y_i - \hat{y}_{(i)}}{\hat{\sigma}_{(i)} \left(1 + \underline{x}_i^T \left(\mathbf{X}_{(i)}^T \mathbf{X}_{(i)}\right)^{-1} \underline{x}_i\right)^{1/2}}$$

Donde $\hat{y}_{(i)} = \underline{x}_i^t \underline{\hat{\beta}_{(i)}}$ es el valor ajustado de la $i-$ésima observación quitando esa misma observación del ajuste, por lo que $\underline{\hat{\beta}_{(i)}}$ y $\hat{\sigma}_{(i)}$ son los estimadores de $\underline{\beta}$ y σ^2 quitando la i-ésima observación. De la misma forma $\mathbf{X}_{(i)}$ es la matriz de diseño sin el $i-$ésimo renglón. Afortunadamente existe una forma fácil de calcular este residual por medio del residual studentizado.

$$t_i = r_i \left(\frac{n-p-1}{n-p-r_i^2}\right)^{1/2}$$

La gran ventaja de esta última fórmula es que no requiere estar ajustando n regresiones.

Finalmente, se puede probar que bajo el supuesto de que $\underline{\varepsilon} \sim N_n\left(\underline{0}, \sigma^2 \mathbf{I}_n\right)$ se tiene que:

$$t_i \sim t_{n-p-1}$$

Luego entonces, para detectar un outlier se suele utilizar la corrección de Bonferroni y comparar contra el cuantil α/n de una distribución t. Por ejemplo si $\alpha = 0.05$ entonces se dice que la i-ésima observación es outlier si

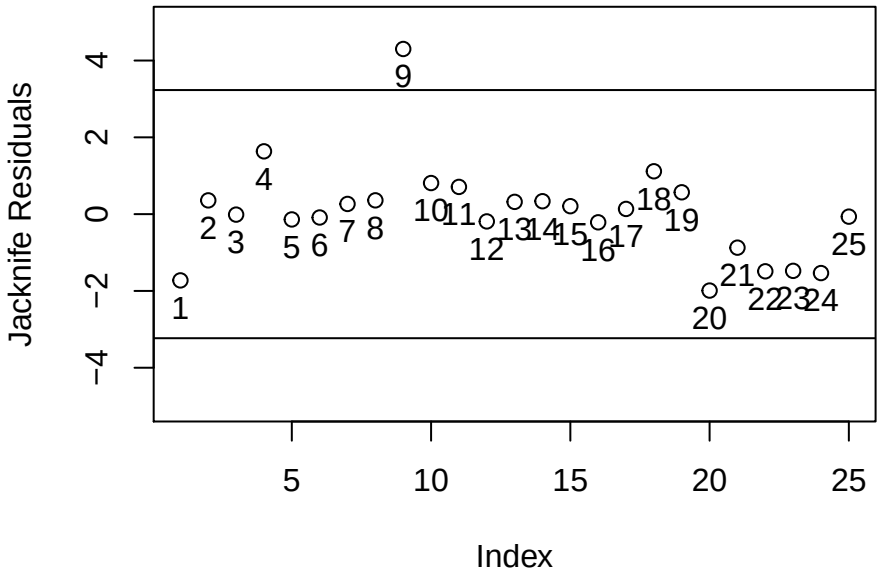
$$|t_i| > t_{n-p-1}^{(1-\alpha^*)} \qquad con \quad \alpha^* = 0.05/(2n)$$

Notas:

- Dos o mas outliers juntos pueden ocultarse entre ellos.
- Un outlier en un modelo puede dejar de serlo en otro cuando hubieron transformaciones, se recomienda hacer el análisis de outliers cada vez que se hace alguna transformación al modelo

En R, podemos obtener los residuales de *jackknife* mediante la función de R rstudent.

```
rj = rstudent(model)
# Graficamos residuales (Jacknife)
plot(rj, xlab = "Index", ylab = "Jacknife Residuals", ylim = c(-5, 5))
text(rj, pos = 1)
alpha = 0.05
# Construimos la las bandas de deteccion de outlier construidas a partir de
# la prueba T, correjida por el factor de Bonferroni
abline(h = c(qt(1 - alpha/n, n - p - 2), -qt(1 - alpha/n, n - p - 2)))
```



Luego entonces, bajo esta metodología la observación 9 es un outlier.

1.2.2. Punto Influyente

Definición 1.2.2 (Punto Influyente) *Un punto influyente es aquel que siendo removido del modelo causa un cambio importante en todo el ajuste.*

Algunas medidas para detectar la influencia de una observación son:

- Cambio en el vector de los coeficientes ajustados: $\underline{\hat{\beta}} - \underline{\hat{\beta}}_{(i)}$
- Cambio en el vector de observaciones $\mathbf{X}^t\left(\underline{\hat{\beta}} - \underline{\hat{\beta}}_{(i)}\right) = \underline{\hat{Y}} - \underline{\hat{Y}}_{(i)}$

Surge el problema de definir una distancia que nos ayude a determinar la diferencia entre vectores. Distancia de Cook:

$$D_i = \frac{\left(\underline{\hat{\beta}} - \underline{\hat{\beta}}_{(i)}\right)^T \left(\mathbf{X}^T\mathbf{X}\right)\left(\underline{\hat{\beta}} - \underline{\hat{\beta}}_{(i)}\right)}{(p+1)\hat{\sigma}^2}$$

$$D_i = \frac{1}{p+1} r_i^2 \frac{h_i}{1-h_i}$$

Un regla usada para la detección de observaciones influyentes es considerar $D_i > 1$ (Cook, R. Dennis; and Weisberg, Sanford (1982)). Pero se ha visto que es muy conservadora, algunos entonces utilizan $D_i > 4/n$ (Bollen, Kenneth A.; and Jackman, Robert W. (1990);)

Por otro lado, Belsey, Kuh y Welch introducen otras dos medias de influencia. La primera es una estadiística que indica cuanto cambia el coeficiente de regresión $\hat{\beta}_j$ cuando la i-ésima observación es removida.

$$DFBETA_{j,i} = \frac{\hat{\beta}_j - \hat{\beta}_{j(i)}}{\sqrt{\hat{\sigma}^2_{(i)} C_{jj}}}$$

Donde C_{jj} es el j-ésimo elemento de la matriz $\left(\mathbf{X}^t\mathbf{X}\right)^{-1}$. Luego entonces un valor (en magnitud) grande de $DFBETA_{j,i}$ indica que la observación i tiene un influencia sobre el coeficiente j Los autores sugieren un punto de corte igual a $2/\sqrt{n}$

$$|DFBETA_{j,i}| > 2/\sqrt{n}$$

También podemos medir la influencia de una observación sobre los valores ajustados $\hat{y}_i$. El diagnostico propuesto es:

$$DFFITT_i = \frac{\hat{y}_i - \hat{y}_{(i)}}{\sqrt{\hat{\sigma}^2_{(i)} h_{ii}}}$$

Donde $\hat{y}_{(i)}$ es el valor ajustado para y_i, obtenido sin usa la i-ésima observación. El denominador no es mas que una estandarización pues se puede probar que $var\left(\hat{y}_i\right) = \sigma^2 h_{ii}$. (Recuerde que h_{ii} son

los elementos de la diagonal de la matriz Sombrero).
La interpretación para el $DFFITT$ es que mide la cantidad de desviaciones estándar que cambia el valor ajustado $\hat{y}_i$ si se elimina la observación i. Obviamente un valor grande (magnitud) nos indica una influencia de la observación i. Los autores sugieren que merece investigarse toda observación tal que:

$$|DFFITS_i| > 2\sqrt{p/n}$$

Los diagnósitos que hemos visto permiten conocer el efecto de las observaciones sobre los coeficientes y las estimaciones y no proporcionan información sobre la *precisión* de la estimación. Recordemos que la teoría inferencial nos indica que un estimador es mejor que entro si este último tiene una menor varianza, luego entonces resultaría util saber si quitando una observación la estimación mejora.
Surge entonces la necesidad de tener una medida escalar de la varianza de de un vector. Lo que se estila utilizar como una medida escalar es utilizar el determinante de la matriz de varianzas y covarianzas . Definimos entonces

Definición 1.2.3 (Varianza Generalizada)

$$GV\left(\underline{\hat{\beta}}\right) = \left|var\left(\underline{\hat{\beta}}\right)\right| = \left|\sigma^2\left(\boldsymbol{X}^t\boldsymbol{X}\right)^{-1}\right|$$

Finalmente para tener una medida de cuanta precisión ganamos o perdemos por quitar una observación definimos lo siguiente:

$$COVRATIO_i = \frac{\left|\left(\mathbf{X}_{(i)}^t\mathbf{X}_{(i)}\right)^{-1}\sigma_{(i)}^2\right|}{\left|\left(\mathbf{X}^t\mathbf{X}\right)^{-1}\sigma^2\right|}$$

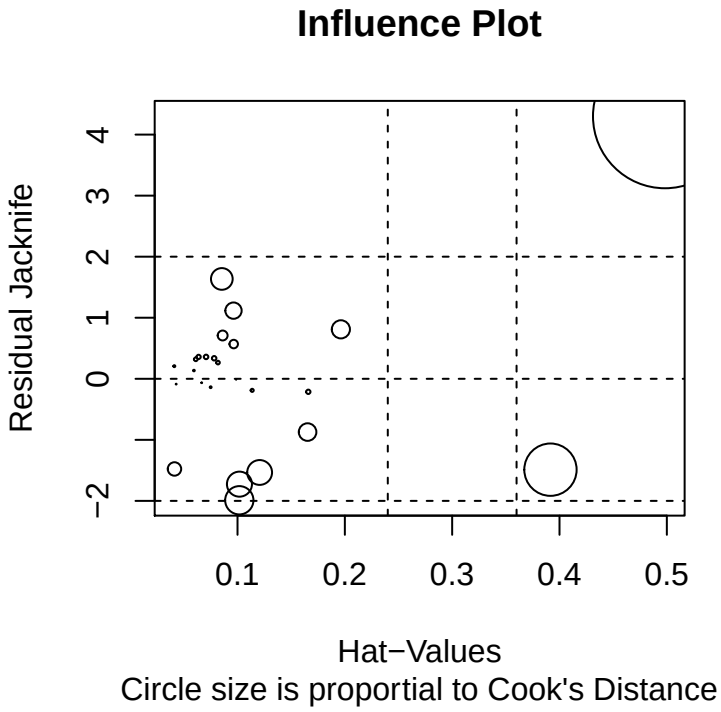
Queda claro entonces que un $COVRATIO_i < 1$ indica que la i-ésima observación mejora la precisión de la estimación. No es fácil obtener valores de corte para este índice pero Belsley, Kuh y Welsh (1980) sugieren que si:

$$COVRATIO_i > 1 + 3p/n \quad o \quad COVRATIO_i < 1 - 3p/n$$

Entonces se debe considerar al punto i como influyente. (Estos valores solo se recomiendan para muestras grandes).

En R estas medidas para la influencia de observaciones son calculadas por la función *influence.measures(model)*

```r
# Funcion en R que obtiene las estadisticas para medir la influencia
influence.measures(model)

## Influence measures of
##   lm(formula = y ~ ., data = DATA) :
##
##      dfb.1_  dfb.cajs dfb.dstn    dffit cov.r   cook.d    hat inf
## 1  -0.19061  0.418659 -0.44263 -0.58105 0.860 1.03e-01 0.1018
## 2   0.09006 -0.047904  0.01446  0.09891 1.215 3.40e-03 0.0707
## 3  -0.00287  0.003222 -0.00232 -0.00425 1.276 6.30e-06 0.0987
## 4   0.45095  0.088082 -0.27276  0.49968 0.877 7.73e-02 0.0854
## 5  -0.03158 -0.013261  0.02417 -0.03934 1.240 5.40e-04 0.0750
## 6  -0.01445  0.001764  0.00106 -0.01848 1.200 1.19e-04 0.0429
## 7   0.07828 -0.022338 -0.01105  0.07920 1.240 2.18e-03 0.0818
## 8   0.07109  0.033331 -0.05374  0.09362 1.206 3.04e-03 0.0637
## 9  -2.56964  0.926545  1.50398  4.28591 0.345 3.41e+00 0.4983   *
## 10  0.10832 -0.339429  0.34260  0.40021 1.305 5.42e-02 0.1963
## 11 -0.03424  0.092440 -0.00268  0.21775 1.172 1.62e-02 0.0861
## 12 -0.03034 -0.048785  0.05410 -0.06783 1.291 1.60e-03 0.1137
## 13  0.07260 -0.035738  0.01137  0.08152 1.207 2.31e-03 0.0611
## 14  0.04982 -0.067492  0.06219  0.09795 1.227 3.33e-03 0.0782
## 15  0.02239 -0.004813  0.00687  0.04279 1.192 6.38e-04 0.0411
## 16 -0.00264  0.063077 -0.08243 -0.09513 1.370 3.15e-03 0.1659
## 17  0.02890  0.006499 -0.01572  0.03397 1.219 4.03e-04 0.0594
## 18  0.24789  0.189220 -0.27169  0.36432 1.070 4.38e-02 0.0963
## 19  0.17227  0.023534 -0.09880  0.18585 1.215 1.19e-02 0.0964
## 20  0.16747 -0.214275 -0.09260 -0.66952 0.762 1.32e-01 0.1017
## 21 -0.16187 -0.297067  0.33628 -0.38836 1.238 5.08e-02 0.1653
## 22  0.39832 -1.024792  0.57279 -1.19431 1.399 4.51e-01 0.3916   *
## 23 -0.15929  0.037161 -0.05247 -0.30645 0.891 2.97e-02 0.0413
## 24 -0.11911  0.402569 -0.46308 -0.56824 0.951 1.01e-01 0.1206
## 25 -0.01656  0.000837  0.00551 -0.01736 1.231 1.05e-04 0.0666

# Funcion en R que grafica las estadisticas para medir la influencia
influencePlot(model, id.method = "identify", ylab = "Residual Jacknife", main = "Influence Plot",
    sub = "Circle size is proportial to Cook's Distance")
```


Influence Plot
Residual Jacknife
4
3
2
1
0
−2
0.1
0.2
0.3
0.4
0.5
Hat−Values
Circle size is proportial to Cook's Distance

Capítulo 2

Selección de modelos

2.1. Metodos de selección de variables

Hasta ahora hemos venido trabajando suponiendo que contamos con todas las variables explicativas que influyen en el modelo. Por lo general esta situación solo ocurre en contadas ocasiones donde la experiencia previa del analista son útiles para seleccionar las variables que deben de estar ajustando en el modelo, sin embargo, en la mayora de los casos , el analísta tiene una variedad de variables candidatas que podrían o no influir en la variable respuesta.
Al problema de encontrar un subconjunto adecuado de variables explicativas se le conoce como el *el problema de selección de variables*

El algoritmo general para llevar a cabo la selección del modelo se puede resumir en lo siguiente:

- Se emplea un criterio de selección de variables en particular
- El modelo resultante se revisa para verificar que las especificaciones funcionales sean correctas (Análisis de Residuales) y que no existan outliers ni observaciones de alta influencia.
- Si el modelo no pasa el punto anterior, se debe de repetir el proceso de selección de variables omitiendo el modelo resultante de las iteraciones anteriores.

Debe de quedar claro que ninguno de los procedimientos garantizarán que se encuentre la mejor ecuación de regresión, además el analísta no debe de confiar demasiado en los resultados de un procedimiento en particular de selección de variables. La experiencia previa del analista, juicios personales siempre deben de entrar en el problema de selección de la mejor ecuación de regresión.

A continuación se presentan criterios para evaluar y comprar modelos de regresión, debe de quedar claro que todos estos modelos deben de ser submodelos (modelos incompletos) de un modelo general saturado:

- Coeficiente de Determiandción Múltiple. R^2
- R^2 Ajustado (Penaliza la entrada de variables)
- SCE (Suma de cuadrados del error)

Cuando el número de variables lo permite p pequeña (< 30) lo que se puede hacer es realizar un análisis exhaustivo de todos los modelos posibles: $2^{30} - 1 = 1,073,741,824$ y escoger el mejor bajo algún criterio.

Los métodos anteriores tiene el problema que no nos dan un punto de corte que decida con cuantas variables ajustemos. Para resolver este problema en la literatura existen algoritmos que determinan el mejor modelo con base en ciertas pruebas estadísticas. Los métodos mas utilizados son:

- Forward
- Backward
- Stepwise

Ojo: Los algoritmos no siempre garantizan encontrar el mejor modelo además de que cada metodología puede arrojar distintas soluciones. El analísta no debe de confiar demasiado en los resultados de un procedimiento en particular de selección de variables. La experiencia previa del analista y juicios personales siempre deben de entrar en el problema de selección de la mejor ecuación de regresión.

2.1.1. Método Forward

El método Forward (o de selección hacia adelante): Inicialmente no hay ninguna variable seleccionada. Se comienza eligiendo la variable que más correlacionada está con la variable respuesta , y si su p-value es pequeño (menor a α) entonces se queda en el modelo en caso de que no sea significativa el algoritmo termina. A continuación selecciona la segunda variable que junto con la primera tiene un valor de la estadística F mas grande, lo que se traduce en un incremento grande en la SCR. El algoritmo finaliza cuando ninguna de las variables no seleccionadas agrega un incremento significativo en la SCR. Ejemplo:

$$H_0 : y = \beta_0 + \beta_k x_{ki} + \varepsilon_i \quad vs \quad H_1 : \beta_0 + \beta_k x_{ki} + \beta_j x_{ji} + \varepsilon_i \quad j \in \{1, \ldots, p\} - \{k\}$$

$$F = \frac{\frac{SCR_{H_1} - SCR_{H_0}}{\sigma^2}}{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sigma^2(n-(2+1))}} = \frac{(SCR_{H_1} - SCR_{H_0})}{\hat{\sigma}^2} \sim \mathscr{F}_{(1,n-(2+1))}$$

2.1.2. Método Backward

El método Backward (o de eliminación hacia atrás) actúa de forma inversa. Se comienza seleccionando todas las variables. En cada paso se elimina una variable del modelo, en este caso se decide sacar la que menos significancia tiene, es decir un F pequeño. El algoritmo finaliza cuando quitar alguna otra variable del modelo ocasiona un reducción significativa en la SCR. Ejemplo:

$$H_0 : y = \beta_0 + \beta_1 x_{1i} \ldots \beta_{k-1} x_{(k-1)i} + \beta_{k+1} x_{(k+1)i} + \ldots + \beta_p x_{pi} + \varepsilon_i q$$

$$H_1 : y = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi} + \varepsilon_i$$

$$F = \frac{\frac{SCR_{H_1} - SCR_{H_0}}{\sigma^2}}{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sigma^2(n-(p+1))}} = \frac{(SCR_{H_1} - SCR_{H_0})}{\hat{\sigma}^2} \sim \mathscr{F}_{(1,n-(p+1))}$$

Se elimina la k-ésima variable si el estadístico F es mas chico que el cuantil $\mathscr{F}^{1-\alpha}_{(1,n-(p+1))}$ y si además es el valor F mas pequenño que se obtuvo entre todas las posibles $k \in \{1, \ldots, p\}$

2.1.3. Método Stepwise

El método Stepwise (o regresión por pasos), fue originalmente propuesto por Efroymson en 1960, ha estado disponible en los paquetes estadísticos desde hace muchos años. Este método utiliza una combinación de los dos algoritmos anteriores: en cada paso se introduce o elimina una variable dependiendo de la significación con la que se está trabajando. Debido a la posible correlación de las variables es posible *arrepentirse* de decisiones tomadas en pasos anteriores, bien sea eliminando del conjunto seleccionado la variable introducida en un paso anterior del algoritmo, o bien sea seleccionando una variable previamente eliminada. El algoritmo puede ser inicializado con todas las variables en cuyo caso el primer paso será la eliminación de una variable o bien también se puede inicializar el algoritmo con ninguna variable, en cuyo caso el primer paso siempre será incluir una variable. Es proceso tiene el problema de que en alguna interación se llegue a un ciclo repetido sacando y metiendo la misma variable, en cuyo caso se decide terminar el algoritmo en ese momento.

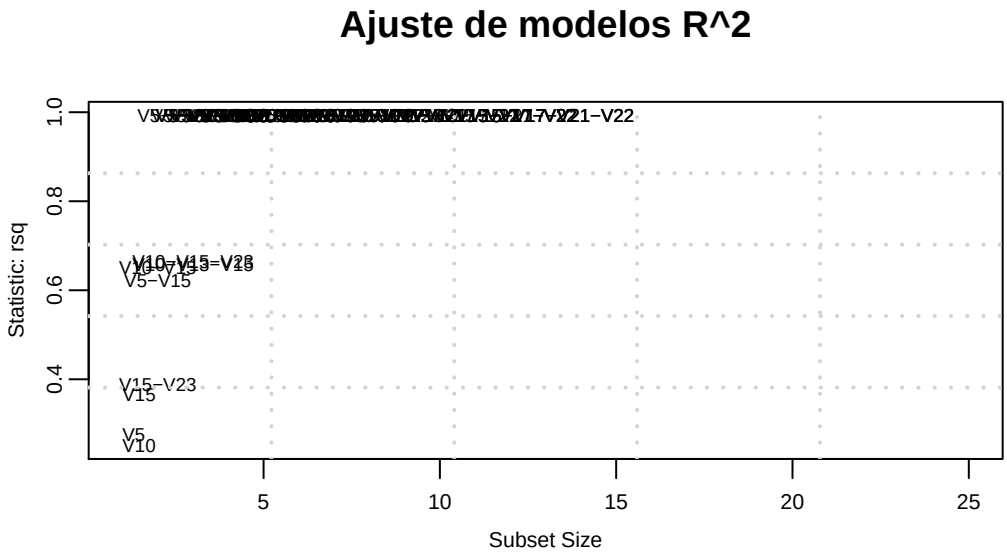
A continuación se presenta el codigo R de la selección de modelos para un ejemplo simulado:

```
# Para ejemplificar el problema de selecci<U+00F3>n de variables simularemos
# un modelo lineal con 25 variables y 200 observaciones.
p = 25
n = 200
set.seed(9)
```

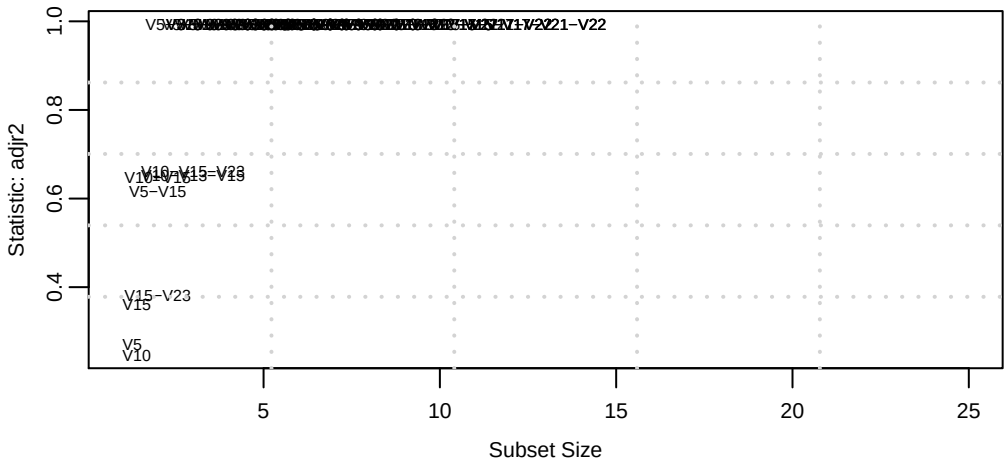
```r
# La matriz de dise<U+00F1>o se simula a traves de variables uniformes
X = round(runif(p * n, 0, 50), 0)
# Generamos matriz de dise<U+00F1>o (multicolinearidad)


X = matrix(X, n, p)
# Verificamos la condicion de nuestra matriz
kappa(t(X) %*% X)

# Generamos Data Frame
X = as.data.frame(X)

# Simulamos Normal
e = rnorm(n, 0, 2)

# Generamos variable respuesta, observe que de las 25 variables solo
# ocupamos las variables: X5, X10, X15 para simular a la variable respueta
# Esperamos que los metodos de seleccion de variables nos ayuden a detectar
# las variables

# OBS: Total de modelos 2^25-1=33,554,431

y = 1 + 1 * X[, 5] + 1 * X[, 10] + 1 * X[, 15] + e

# Aplicaremos los metodos de seleccion

# Generamos Data Frame
Frame = as.data.frame(cbind(y, X))

# Metodo Forward Programa en R
forward <- function(y, X, alpha) {
    p = length(X[1, ])
    v = 0
    select = 1:p
    k = 1
    for (k in 1:p) {
        selection = matrix(0, p - (k - 1), 3)
        j = 1
        if (k == 1) {
            for (i in select) {
                model = lm(y ~ X[, i])
                selection[j, ] = c(i, round(anova(model)$"F value"[1], 3), round(anova(model)$"Pr(>F)"[1],
```

```
                5))
            j = j + 1
        }
    }
    if (k > 1) {
        for (i in select[-v]) {
            model = lm(y ~ X[, v] + X[, i])
            selection[j, ] = c(i, round(anova(model)$"F value"[2], 3), round(anova(model)$"Pr(>F)"[2],
              5))
            j = j + 1
        }
    }
    a = selection[selection[, 2] == max(selection[, 2])]
    if (a[3] < alpha) {
        v = c(v, a[1])
    } else {
        break
    }
  }
  return(v[2:length(v)])
}

X = as.matrix(X)
forward(y, X, 0.05)


# Metodo Forward Programa en R
backward <- function(y, X, alpha) {
  p = length(X[1, ])
  v = 0
  select = 1:p
  k = 1
  for (k in 1:p) {
    selection = matrix(0, p - (k - 1), 3)
    j = 1
    if (k == 1) {
      for (i in select) {
        model = lm(y ~ X[, c(-i)] + X[, i])
        selection[i, ] = c(i, round(anova(model)$"F value"[2], 5), round(anova(model)$"Pr(>F)"[2],
          5))
        j = j + 1
      }
```

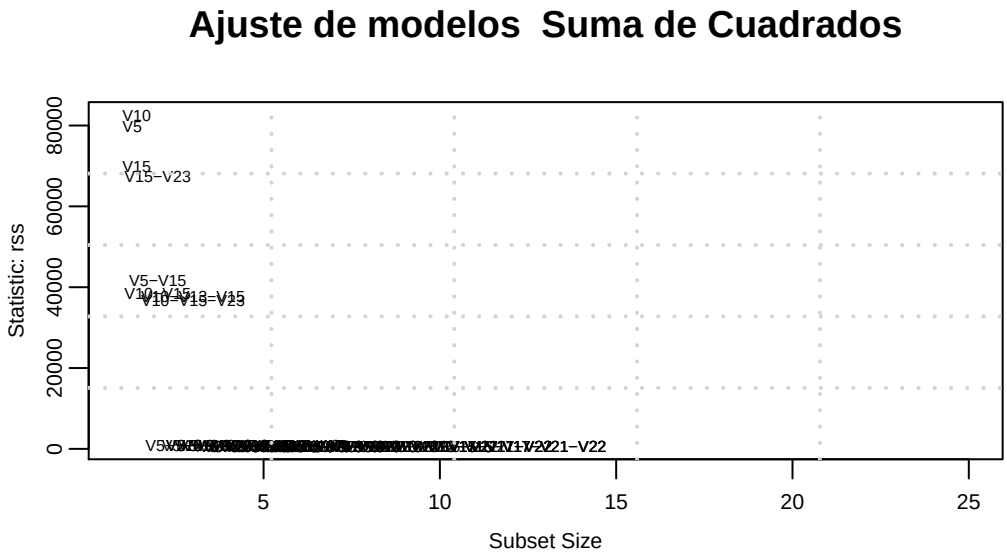
```
        }
        if (k > 1) {
            for (i in select[-v]) {
                model = lm(y ~ X[, -c(v, i)] + X[, i])
                selection[j, ] = c(i, round(anova(model)$"F value"[2], 5), round(anova(model)$"Pr(>F)"[2],
                  5))
                j = j + 1
            }
        }
        a = selection[selection[, 2] == min(selection[, 2])]
        if (a[3] > alpha) {
            v = c(v, a[1])
        } else {
            break
        }
    }
    return(select[-v])
}
backward(y, X, 0.05)

# Otros Metodos implementados en R
library(leaps)
par(mfrow = c(1, 1))
# Funcion leaps busca el mejor modelo usando Forward
leaps <- regsubsets(y ~ ., data = Frame, nbest = 3, really.big = T, nvmax = 10,
    method = "forward")
summary(leaps)
subsets(leaps, legend = FALSE, abbrev = 1, statistic = "rsq", main = "Ajuste de modelos R^2",
    col = 2, cex = 0.6, cex.axis = 0.7, cex.lab = 0.7, mgp = c(1.5, 0.5, 0))
grid(5, 5, lwd = 2)  # grid only in y-direction
```



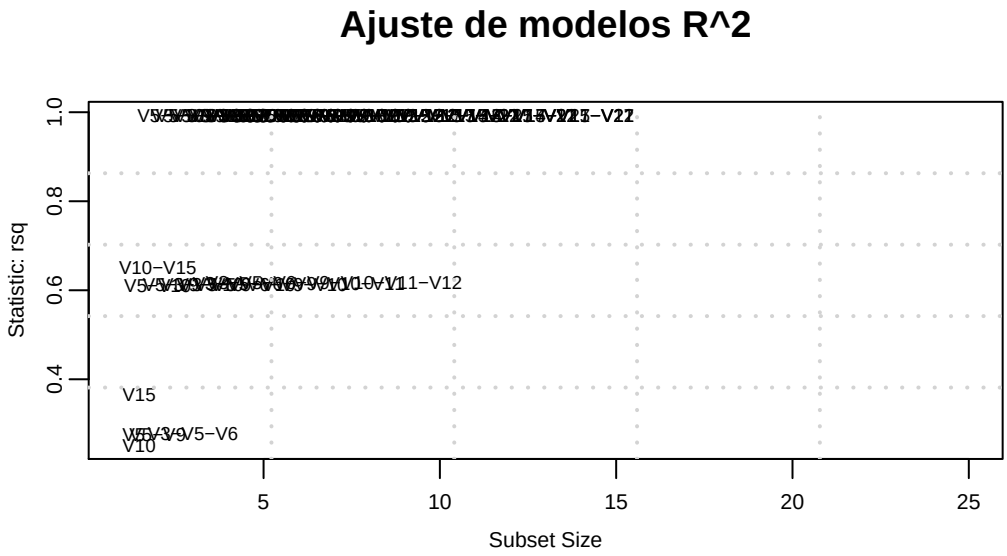

```
subsets(leaps, legend = FALSE, abbrev = 1, statistic = "adjr2", main = "Ajuste de modelos R^2 Ajustado",
    col = 2, cex = 0.5, cex.axis = 0.7, cex.lab = 0.7, mgp = c(1.5, 0.5, 0))
grid(5, 5, lwd = 2)  # grid only in y-direction
```



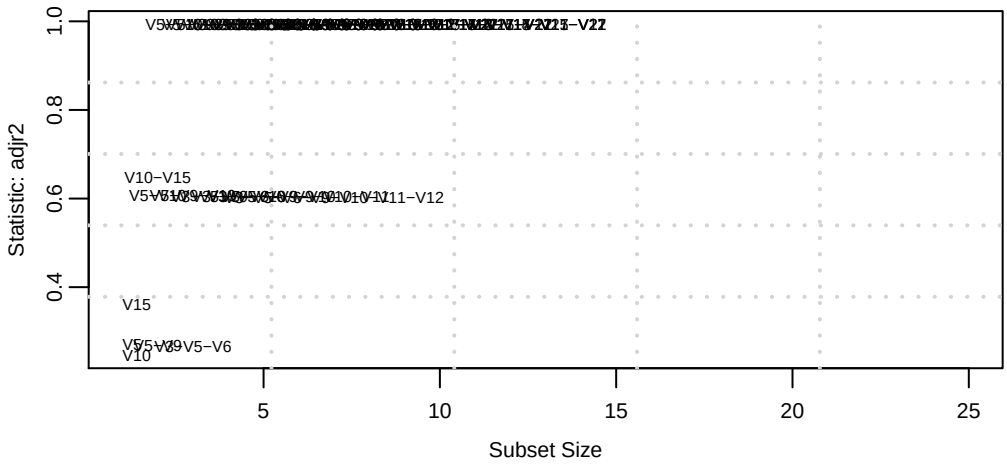
```
subsets(leaps, legend = FALSE, abbrev = 1, statistic = "rss", main = "Ajuste de modelos  Suma de Cuadrados",
    col = 2, cex = 0.5, cex.axis = 0.7, cex.lab = 0.7, mgp = c(1.5, 0.5, 0))
grid(5, 5, lwd = 2)  # grid only in y-direction
```



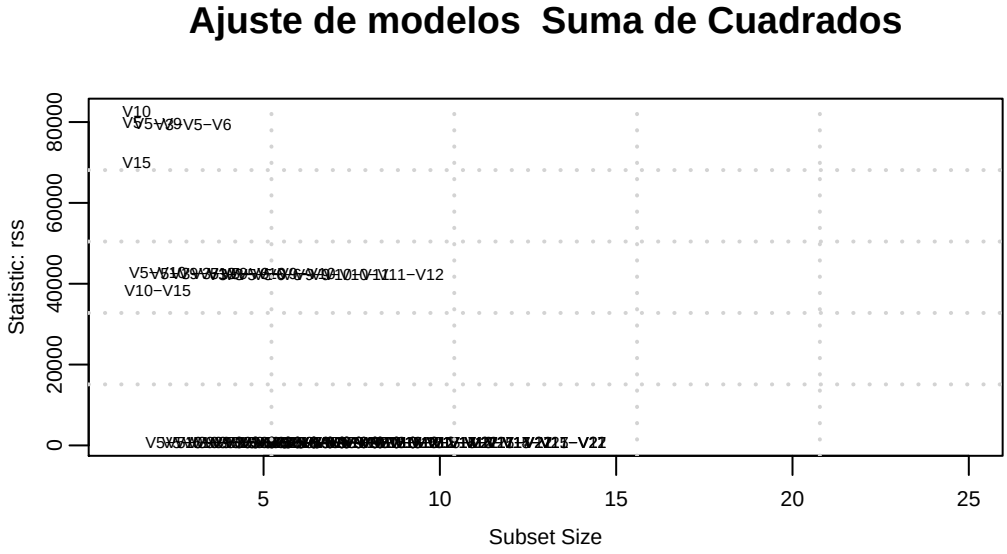

```
# Funcion leaps busca el mejor modelo usando Backward
leaps <- regsubsets(y ~ ., data = Frame, nbest = 3, really.big = T, nvmax = 10,
    method = "backward")
summary(leaps)
subsets(leaps, legend = FALSE, abbrev = 1, statistic = "rsq", main = "Ajuste de modelos R^2",
    col = 2, cex = 0.6, cex.axis = 0.7, cex.lab = 0.7, mgp = c(1.5, 0.5, 0))
grid(5, 5, lwd = 2)  # grid only in y-direction
```




```
subsets(leaps, legend = FALSE, abbrev = 1, statistic = "adjr2", main = "Ajuste de modelos R^2 Ajustado",
    col = 2, cex = 0.5, cex.axis = 0.7, cex.lab = 0.7, mgp = c(1.5, 0.5, 0))
grid(5, 5, lwd = 2)  # grid only in y-direction
```



```
subsets(leaps, legend = FALSE, abbrev = 1, statistic = "rss", main = "Ajuste de modelos  Suma de Cuadrados",
    col = 2, cex = 0.5, cex.axis = 0.7, cex.lab = 0.7, mgp = c(1.5, 0.5, 0))
grid(5, 5, lwd = 2)  # grid only in y-direction
```




```
# Otro Metodo: Todas las posibles regresiones Para ejemplificar el problema
# de selecci<U+00F3>n de variables simularemos un modelo lineal con 6
# variables y 200 observaciones.

p = 6
n = 200
set.seed(9)

# La matriz de dise<U+00F1>o se simula a traves de variables uniformes
X = round(runif(p * n, 0, 50), 0)
# Generamos matriz de dise<U+00F1>o (multicolinearidad)


X = matrix(X, n, p)
# Verificamos la condicion de nuestra matriz
kappa(t(X) %*% X)



# Generamos Data Frame
```

```r
X = as.data.frame(X)

# Simulamos Normal
e = rnorm(n, 0, 2)

# Generamos variable respuesta, observe que de las 6 variables solo ocupamos
# las variables: X2, X4, X6 para simular a la variable respueta Esperamos
# que los metodos de selecci<U+00F3>n de variables nos ayuden a detectar las
# variables

# OBS: Total de modelo 2^6-1=63

y = 1 + 1 * X[, 2] + 1 * X[, 4] + 1 * X[, 6] + e

# Aplicaremos los metodos de selecci<U+00F3>n

# Generamos Data Frame
Frame = as.data.frame(cbind(y, X))
```

```r
# 63 posibles modelos

# Modelos con 1 variable
choose(6, 1)

## [1] 6

# Modelos con 2 variables
choose(6, 2)

## [1] 15

# Modelos con 3 variables
choose(6, 3)

## [1] 20

# Modelos con 4 variables
choose(6, 4)

## [1] 15

# Modelos con 5 variables
choose(6, 5)
```

```
## [1] 6

# Modelos con 6 variables
choose(6, 6)

## [1] 1

leaps <- regsubsets(y ~ ., data = Frame, nbest = 20, really.big = T, nvmax = 10,
    method = "exhaustive")
resumen = summary(leaps)


MODELOS = cbind(resumen$which, R2 = resumen$rsq, AdjR2 = resumen$adjr2, SCE = resumen$rss)
# Lista de los 63 posibles modelos
MODELOS

##   (Intercept) V1 V2 V3 V4 V5 V6        R2     AdjR2      SCE
## 1           1  0  0  0  0  0  1 0.3679515  0.364759  81171.4
## 1           1  0  0  0  1  0  0 0.3445661  0.341256  84174.6
## 1           1  0  1  0  0  0  0 0.3109879  0.307508  88487.0
## 1           1  1  0  0  0  0  0 0.0032342 -0.001800 128010.5
## 1           1  0  0  1  0  0  0 0.0004995 -0.004548 128361.7
## 1           1  0  0  0  0  1  0 0.0004030 -0.004645 128374.1
## 2           1  0  1  0  0  0  1 0.6851358  0.681939  40436.7
## 2           1  0  1  0  1  0  0 0.6677127  0.664339  42674.3
## 2           1  0  0  0  1  0  1 0.6666189  0.663234  42814.7
## 2           1  1  0  0  0  0  1 0.3717303  0.365352  80686.1
## 2           1  0  0  1  0  0  1 0.3684987  0.362088  81101.1
## 2           1  0  0  0  0  1  1 0.3679559  0.361539  81170.8
## 2           1  0  0  1  1  0  0 0.3474501  0.340825  83804.3
## 2           1  1  0  0  1  0  0 0.3457566  0.339115  84021.8
## 2           1  0  0  0  1  1  0 0.3455792  0.338935  84044.5
## 2           1  1  1  0  0  0  0 0.3181293  0.311207  87569.8
## 2           1  0  1  1  0  0  0 0.3121675  0.305184  88335.5
## 2           1  0  1  0  0  1  0 0.3110710  0.304077  88476.3
## 2           1  1  0  1  0  0  0 0.0039564 -0.006156 127917.7
## 2           1  1  0  0  0  1  0 0.0035336 -0.006583 127972.0
## 2           1  0  0  1  0  1  0 0.0008885 -0.009255 128311.7
## 3           1  0  1  0  1  0  1 0.9948633  0.994785    659.7
## 3           1  1  1  0  0  0  1 0.6931334  0.688436  39409.6
## 3           1  0  1  1  0  0  1 0.6862819  0.681480  40289.5
## 3           1  0  1  0  0  1  1 0.6853141  0.680498  40413.8
## 3           1  0  0  1  1  0  1 0.6693846  0.664324  42459.6
```

```
## 3           1  0  1  0  1  1  0 0.6681480  0.663069  42618.4
## 3           1  1  1  0  1  0  0 0.6677723  0.662687  42666.6
## 3           1  0  1  1  1  0  0 0.6677247  0.662639  42672.7
## 3           1  1  0  0  1  0  1 0.6671946  0.662101  42740.8
## 3           1  0  0  0  1  1  1 0.6667242  0.661623  42801.2
## 3           1  1  0  1  0  0  1 0.3725302  0.362926  80583.3
## 3           1  1  0  0  0  1  1 0.3717566  0.362141  80682.7
## 3           1  0  0  1  0  1  1 0.3685048  0.358839  81100.3
## 3           1  0  0  1  1  1  0 0.3484134  0.338440  83680.6
## 3           1  1  0  1  1  0  0 0.3483972  0.338424  83682.6
## 3           1  1  0  0  1  1  0 0.3468921  0.336896  83875.9
## 3           1  1  1  1  0  0  0 0.3189247  0.308500  87467.7
## 3           1  1  1  0  0  1  0 0.3181532  0.307717  87566.7
## 3           1  0  1  1  0  1  0 0.3122596  0.301733  88323.6
## 3           1  1  0  1  0  1  0 0.0042379 -0.011003 127881.6
## 4           1  0  1  1  1  0  1 0.9948888  0.994784    656.4
## 4           1  1  1  0  1  0  1 0.9948729  0.994768    658.5
## 4           1  0  1  0  1  1  1 0.9948642  0.994759    659.6
## 4           1  1  1  1  0  0  1 0.6938812  0.687602  39313.6
## 4           1  1  1  0  0  1  1 0.6934533  0.687165  39368.5
## 4           1  0  1  1  0  1  1 0.6864479  0.680016  40268.2
## 4           1  1  0  1  1  0  1 0.6697975  0.663024  42406.5
## 4           1  0  0  1  1  1  1 0.6694747  0.662695  42448.0
## 4           1  1  1  0  1  1  0 0.6682265  0.661421  42608.3
## 4           1  0  1  1  1  1  0 0.6681620  0.661355  42616.6
## 4           1  1  1  1  1  0  0 0.6677882  0.660974  42664.6
## 4           1  1  0  0  1  1  1 0.6673283  0.660504  42723.6
## 4           1  1  0  1  0  1  1 0.3725626  0.359692  80579.2
## 4           1  1  0  1  1  1  0 0.3494696  0.336125  83544.9
## 4           1  1  1  1  0  1  0 0.3189537  0.304984  87463.9
## 5           1  1  1  1  1  0  1 0.9948964  0.994765    655.4
## 5           1  0  1  1  1  1  1 0.9948896  0.994758    656.3
## 5           1  1  1  0  1  1  1 0.9948742  0.994742    658.3
## 5           1  1  1  1  0  1  1 0.6941841  0.686302  39274.7
## 5           1  1  0  1  1  1  1 0.6699103  0.661403  42392.1
## 5           1  1  1  1  1  1  0 0.6682453  0.659695  42605.9
## 6           1  1  1  1  1  1  1 0.9948975  0.994739    655.3
```

```
subsets(leaps, legend = FALSE, abbrev = 1, statistic = "adjr2", main = "Ajuste de modelos R^2 Ajustado",
    col = 2, cex = 0.5, cex.axis = 0.7, cex.lab = 0.7, mgp = c(1.5, 0.5, 0))
```

```
##    Abbreviation
## V1           V1
```

```
## V2           V2
## V3           V3
## V4           V4
## V5           V5
## V6           V6

grid(5, 5, lwd = 2)  # grid only in y-direction
```