

Comparación de modelos y selección de variables

Contents

1	Introducción	1
2	Comparación de modelos	2
3	Métodos de construcción de modelos a partir de un conjunto de variables	3
3.1	Selección de variables significativas	3
3.2	Best subset selection	5
3.3	Método Forward-Stepwise	9
3.4	Método Backward-Stepwise	10
3.5	Eligiendo el mejor modelo utilizando subconjuntos de validación	12
3.6	Eligiendo el mejor modelo utilizando Cross-Validation	14

1 Introducción

```
d = read.csv("datos/Hitters.csv")
d = d[,-1]
str(d)

## 'data.frame': 322 obs. of 20 variables:
## $ AtBat : int 293 315 479 496 321 594 185 298 323 401 ...
## $ Hits : int 66 81 130 141 87 169 37 73 81 92 ...
## $ HmRun : int 1 7 18 20 10 4 1 0 6 17 ...
## $ Runs : int 30 24 66 65 39 74 23 24 26 49 ...
## $ RBI : int 29 38 72 78 42 51 8 24 32 66 ...
## $ Walks : int 14 39 76 37 30 35 21 7 8 65 ...
## $ Years : int 1 14 3 11 2 11 2 3 2 13 ...
## $ CAtBat : int 293 3449 1624 5628 396 4408 214 509 341 5206 ...
## $ CHits : int 66 835 457 1575 101 1133 42 108 86 1332 ...
## $ CHmRun : int 1 69 63 225 12 19 1 0 6 253 ...
## $ CRuns : int 30 321 224 828 48 501 30 41 32 784 ...
## $ CRBI : int 29 414 266 838 46 336 9 37 34 890 ...
## $ CWalks : int 14 375 263 354 33 194 24 12 8 866 ...
## $ League : chr "A" "N" "A" "N" ...
## $ Division : chr "E" "W" "W" "E" ...
## $ PutOuts : int 446 632 880 200 805 282 76 121 143 0 ...
## $ Assists : int 33 43 82 11 40 421 127 283 290 0 ...
## $ Errors : int 20 10 14 3 4 25 7 9 19 0 ...
## $ Salary : num NA 475 480 500 91.5 750 70 100 75 1100 ...
## $ NewLeague: chr "A" "N" "A" "N" ...
```

Comprobamos si hay missing obsevation (NA) en el salario:

```
sum(is.na(d$Salary))
```

```
## [1] 59
```

Eliminamos estos datos:

```
d = na.omit(d)
```

2 Comparación de modelos

Se pueden utilizar las siguientes métricas para comparar modelos:

- R-cuadrado
- Residual Sum of Squares, RSS: $\text{sum}((\text{observed} - \text{predicted})^2)$.
- Mean Squared Error, MSE = $\text{mean}((\text{observed} - \text{predicted})^2)$. Cuanto menor sea el MSE, mejor.
- Residual Standard Error, RSE = $\text{sum}((\text{observed} - \text{predicted})^2)/(n-k-1)$.
- Mean Absolute Error, MAE = $\text{mean}(\text{abs}(\text{observed} - \text{predicted}))$.

El problema con estas métricas es que dependen del número de regresores considerados. Por tanto se pueden utilizar para **comparar modelos con el mismo número de regresores**. Otras métricas que no tienen este problema son:

- Akaike Information criteria:

$$AIC = \frac{1}{n\hat{\sigma}^2}(RSS + 2k\hat{\sigma}^2)$$

- Estadístico Cp de Mallows:

$$Cp = \frac{1}{n}(RSS + 2k\hat{\sigma}^2)$$

Cp y AIC son proporcionales, $Cp = AIC * \hat{\sigma}^2$.

- Bayesian Information Criteria:

$$BIC = \frac{1}{n}(RSS + \log(n)k\hat{\sigma}^2)$$

- R-cuadrado ajustado:

$$R^2 - \text{ajustado} = 1 - \frac{RSS/(n-k-1)}{TSS/(n-1)}$$

donde:

- k: número de regresores.
- $\hat{\sigma}^2$: estimación del error del modelo, la varianza residual.
- RSS: Residual sum of squares

$$RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- TSS: Total Sum of Squares

$$TSS = \sum_{i=1}^n (y_i - \bar{y})^2$$

Por último, se puede utilizar el método del subconjunto de validación y el de validación cruzada para comparar modelos, sobre todo **desde un punto de vista predictivo**.

3 Métodos de construcción de modelos a partir de un conjunto de variables

3.1 Selección de variables significativas

Algoritmo:

1. M_p es el modelo con todos los regresores.
2. Para $k = p, \dots, 1$
 - a. Se estima el modelo con k regresores, M_k .
 - b. Se elimina la variable con mayor pvalor de los contrastes individuales.
3. Elegir el modelo con el mayor número de regresores significativos.

```
m_1 = lm(Salary ~ ., data = d)
summary(m_1)

##
## Call:
## lm(formula = Salary ~ ., data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -907.62 -178.35  -31.11  139.09 1877.04
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  163.10359    90.77854   1.797  0.073622 .
## AtBat        -1.97987     0.63398  -3.123  0.002008 **
## Hits         7.50077     2.37753   3.155  0.001808 **
## HmRun         4.33088     6.20145   0.698  0.485616
## Runs        -2.37621     2.98076  -0.797  0.426122
## RBI          -1.04496     2.60088  -0.402  0.688204
## Walks         6.23129     1.82850   3.408  0.000766 ***
## Years        -3.48905    12.41219  -0.281  0.778874
## CAtBat       -0.17134     0.13524  -1.267  0.206380
## CHits         0.13399     0.67455   0.199  0.842713
## CHmRun       -0.17286     1.61724  -0.107  0.914967
## CRuns         1.45430     0.75046   1.938  0.053795 .
## CRBI          0.80771     0.69262   1.166  0.244691
## CWalks       -0.81157     0.32808  -2.474  0.014057 *
## LeagueN      62.59942    79.26140   0.790  0.430424
## DivisionW    -116.84925   40.36695  -2.895  0.004141 **
## PutOuts       0.28189     0.07744   3.640  0.000333 ***
## Assists       0.37107     0.22120   1.678  0.094723 .
## Errors       -3.36076     4.39163  -0.765  0.444857
## NewLeagueN   -24.76233    79.00263  -0.313  0.754218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 315.6 on 243 degrees of freedom
## Multiple R-squared:  0.5461, Adjusted R-squared:  0.5106
## F-statistic: 15.39 on 19 and 243 DF,  p-value: < 2.2e-16

m_2 = lm(Salary ~ . - CHmRun, data = d)
summary(m_2)
```

```
##
## Call:
## lm(formula = Salary ~ . - CHmRun, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -914.51 -175.66  -31.72  137.49 1876.79
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  163.08380   90.59426   1.800 0.073071 .
## AtBat        -1.97939    0.63268  -3.129 0.001970 **
## Hits         7.44499    2.31485   3.216 0.001475 **
## HmRun        4.03304    5.52892   0.729 0.466429
## Runs        -2.27127    2.80872  -0.809 0.419504
## RBI         -0.96237    2.47840  -0.388 0.698131
## Walks        6.20550    1.80884   3.431 0.000707 ***
## Years       -3.42721   12.37355  -0.277 0.782031
## CAtBat       -0.17461    0.13146  -1.328 0.185336
## CHits        0.18359    0.48861   0.376 0.707440
## CRuns        1.40160    0.56455   2.483 0.013714 *
## CRBI         0.73870    0.25026   2.952 0.003468 **
## CWalks       -0.80172    0.31424  -2.551 0.011343 *
## LeagueN      63.12305   78.94944   0.800 0.424756
## DivisionW   -116.85917  40.28499  -2.901 0.004062 **
## PutOuts      0.28224    0.07721   3.655 0.000315 ***
## Assists      0.37319    0.21986   1.697 0.090902 .
## Errors      -3.38913    4.37472  -0.775 0.439262
## NewLeagueN  -25.31356   78.67426  -0.322 0.747916
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 314.9 on 244 degrees of freedom
## Multiple R-squared:  0.5461, Adjusted R-squared:  0.5126
## F-statistic: 16.31 on 18 and 244 DF,  p-value: < 2.2e-16
```

```
m_3 = lm(Salary ~ . - CHmRun - Years, data = d)
summary(m_3)
```

```
##
## Call:
## lm(formula = Salary ~ . - CHmRun - Years, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -912.02 -180.92  -34.89  138.05 1881.51
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  148.43333   73.41097   2.022 0.044268 *
## AtBat        -1.95091    0.62309  -3.131 0.001953 **
## Hits         7.39141    2.30241   3.210 0.001503 **
## HmRun        4.08280    5.51558   0.740 0.459869
## Runs        -2.23967    2.80111  -0.800 0.424737
## RBI         -0.99402    2.47109  -0.402 0.687845
```

```
## Walks          6.19706      1.80517      3.433 0.000701 ***
## CAtBat        -0.19133      0.11657     -1.641 0.102010
## CHits          0.20673      0.48050      0.430 0.667398
## CRuns          1.42497      0.55716      2.558 0.011144 *
## CRBI           0.74147      0.24958      2.971 0.003265 **
## CWalks        -0.80376      0.31356     -2.563 0.010966 *
## LeagueN        64.19282     78.70619      0.816 0.415521
## DivisionW     -116.06176     40.10620     -2.894 0.004148 **
## PutOuts         0.28303      0.07702      3.675 0.000292 ***
## Assists         0.37732      0.21894      1.723 0.086083 .
## Errors         -3.31999      4.35935     -0.762 0.447044
## NewLeagueN    -24.88922     78.51099     -0.317 0.751502
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 314.3 on 245 degrees of freedom
## Multiple R-squared:  0.546, Adjusted R-squared:  0.5144
## F-statistic: 17.33 on 17 and 245 DF,  p-value: < 2.2e-16
```

Y así sucesivamente. Este método se suele utilizar cuando utilizamos el modelo para explicar relaciones entre variables, por lo que estamos interesados en variables significativas. Cuando el objetivo es predecir se utilizan los métodos que se indican a continuación.

3.2 Best subset selection

Algoritmo:

Para $k = 1, 2, \dots, p$:

- Estimar todos los modelos de k regresores (hay $\binom{p}{k}$ modelos posibles).
- Elegir el que tenga menor RSS o mayor R^2 . Este será el modelo M_k .

```
library(leaps)
m2 = regsubsets(Salary ~ ., data = d)
summary(m2)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = d)
## 19 Variables (and intercept)
##              Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun           FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
```

```

## Assists      FALSE      FALSE
## Errors       FALSE      FALSE
## NewLeagueN   FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##           AtBat Hits HmRun Runs RBI Walks Years CatBat CHits CHmRun CRuns CRBI CWalks LeagueN Division
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " "
## 2 ( 1 ) " " "*" " " " " " " " " " " " " " " "*" " " " "
## 3 ( 1 ) " " "*" " " " " " " " " " " " " " " "*" " " " "
## 4 ( 1 ) " " "*" " " " " " " " " " " " " " " "*" " " "*"
## 5 ( 1 ) "*" "*" " " " " " " " " " " " " " " "*" " " "*"
## 6 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " "*" " " "*"
## 7 ( 1 ) " " "*" " " " " " " "*" " " "*" "*" "*" " " " " "*"
## 8 ( 1 ) "*" "*" " " " " " " "*" " " " " "*" "*" "*" " " "*"
##           Assists Errors NewLeagueN
## 1 ( 1 ) " " " " " "
## 2 ( 1 ) " " " " " "
## 3 ( 1 ) " " " " " "
## 4 ( 1 ) " " " " " "
## 5 ( 1 ) " " " " " "
## 6 ( 1 ) " " " " " "
## 7 ( 1 ) " " " " " "
## 8 ( 1 ) " " " " " "

```

El resultado son los mejores 8 modelos (por defecto):

- la primera línea es el mejor modelo (en términos de R^2) de una variable. La variable seleccionada es la que aparece con un asterisco, **CRBI**.
- la segunda línea es el mejor modelo (en términos de R^2) de dos variables, **Hits** y **CRBI**.
- y así sucesivamente.

Podemos seleccionar el número de modelos que nos devuelve con *nvmax*:

```

m_best = regsubsets(Salary ~ ., data = d, nvmax = 19)
summary(m_best)

```

```

## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = d, nvmax = 19)
## 19 Variables (and intercept)
##           Forced in Forced out
## AtBat      FALSE      FALSE
## Hits       FALSE      FALSE
## HmRun      FALSE      FALSE
## Runs       FALSE      FALSE
## RBI        FALSE      FALSE
## Walks      FALSE      FALSE
## Years      FALSE      FALSE
## CatBat     FALSE      FALSE
## CHits      FALSE      FALSE
## CHmRun     FALSE      FALSE
## CRuns      FALSE      FALSE
## CRBI       FALSE      FALSE
## CWalks     FALSE      FALSE
## LeagueN    FALSE      FALSE
## DivisionW  FALSE      FALSE
## PutOuts    FALSE      FALSE

```

```

## Assists          FALSE      FALSE
## Errors           FALSE      FALSE
## NewLeagueN       FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: exhaustive
##      AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks LeagueN Divisi
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " "
## 2 ( 1 ) " " "*" " " " " " " " " " " " " " " " " " "
## 3 ( 1 ) " " "*" " " " " " " " " " " " " " " " " "
## 4 ( 1 ) " " "*" " " " " " " " " " " " " " " " " "
## 5 ( 1 ) "*" "*" " " " " " " " " " " " " " " " " "
## 6 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " " "
## 7 ( 1 ) " " "*" " " " " " " "*" " " "*" "*" " " " " " "
## 8 ( 1 ) "*" "*" " " " " " " "*" " " " " "*" " " "*" " "
## 9 ( 1 ) "*" "*" " " " " " " "*" " " "*" " " "*" "*" " "
## 10 ( 1 ) "*" "*" " " " " " " "*" " " "*" " " "*" "*" "*" "
## 11 ( 1 ) "*" "*" " " " " " " "*" " " "*" " " "*" "*" "*" "*"
## 12 ( 1 ) "*" "*" " " " " "*" " " "*" " " "*" "*" "*" "*" "*"
## 13 ( 1 ) "*" "*" " " " " "*" " " "*" " " "*" "*" "*" "*" "*"
## 14 ( 1 ) "*" "*" "*" "*" " " "*" " " "*" " " "*" "*" "*" "*" "*"
## 15 ( 1 ) "*" "*" "*" "*" " " "*" " " "*" " " "*" "*" "*" "*" "*"
## 16 ( 1 ) "*" "*" "*" "*" "*" "*" " " "*" " " "*" " " "*" "*" "*" "*"
## 17 ( 1 ) "*" "*" "*" "*" "*" "*" " " "*" " " "*" " " "*" "*" "*" "*"
## 18 ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" " " "*" " " "*" "*" "*" "*"
## 19 ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" " " "*" "*" "*" "*"
##      Assists Errors NewLeagueN
## 1 ( 1 ) " " " " "
## 2 ( 1 ) " " " " "
## 3 ( 1 ) " " " " "
## 4 ( 1 ) " " " " "
## 5 ( 1 ) " " " " "
## 6 ( 1 ) " " " " "
## 7 ( 1 ) " " " " "
## 8 ( 1 ) " " " " "
## 9 ( 1 ) " " " " "
## 10 ( 1 ) "*" " " "
## 11 ( 1 ) "*" " " "
## 12 ( 1 ) "*" " " "
## 13 ( 1 ) "*" "*" "
## 14 ( 1 ) "*" "*" "
## 15 ( 1 ) "*" "*" "
## 16 ( 1 ) "*" "*" "
## 17 ( 1 ) "*" "*" "*"
## 18 ( 1 ) "*" "*" "*"
## 19 ( 1 ) "*" "*" "*"

```

Podemos trabajar con R2 o con las otras métricas:

```

m_best_summary = summary(m_best)
names(m_best_summary)

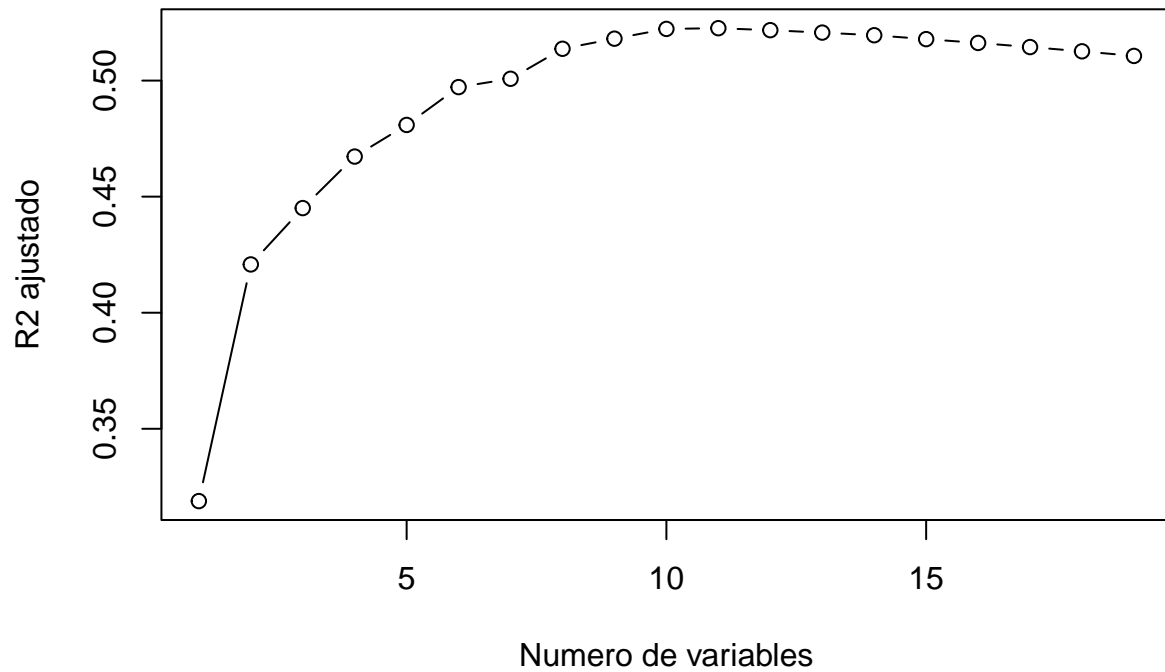
```

```

## [1] "which" "rsq" "rss" "adjr2" "cp" "bic" "outmat" "obj"

```

```
plot(m_best_summary$adjr2, xlab = "Numero de variables", ylab = "R2 ajustado", type = "b")
```



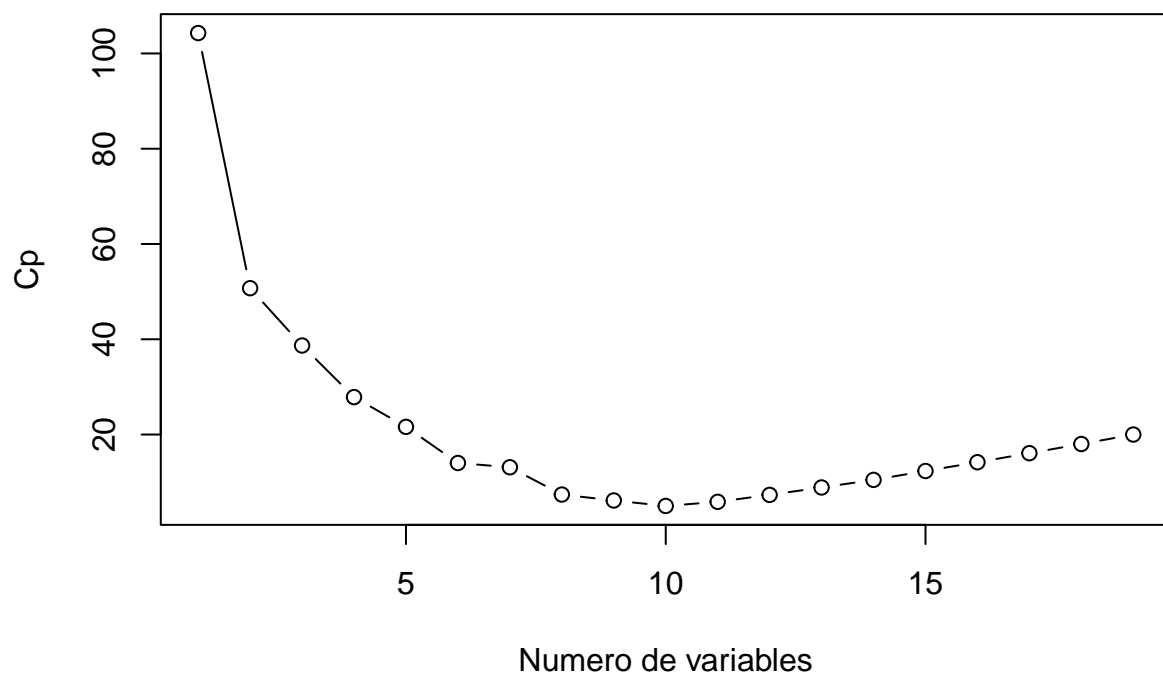
Buscamos el máximo:

```
which.max(m_best_summary$adjr2)
```

```
## [1] 11
```

Si utilizamos el criterio del Cp (que es equivalente al AIC):

```
plot(m_best_summary$cp, xlab = "Numero de variables", ylab = "Cp", type = "b")
```




```
which.min(m_best_summary$cp)
```

```
## [1] 10
```

Justificación: el RSS disminuye con el número de regresores k . Por eso penalizamos incluyendo un término que contiene a k .

Los coeficientes estimados con ese modelo son:

```
coef(m_best,10)
```

```
## (Intercept)      AtBat      Hits      Walks      CAtBat      CRuns      CRBI      CWalks
## 162.5354420 -2.1686501  6.9180175  5.7732246 -0.1300798  1.4082490  0.7743122 -0.8308
## DivisionW     PutOuts     Assists
## -112.3800575  0.2973726  0.2831680
```

3.3 Método Forward-Stepwise

Algoritmo:

1. M_0 es el modelo sin regresores.
2. Para $k = 0, \dots, (p-1)$
 - a. A partir del modelo con k regresores, M_k , estimar todos los modelos posibles con $(k+1)$ regresores.
 - b. Elegir el que tenga menor RSS o mayor R^2 . Este será el modelo M_{k+1} .
3. Elegir el mejor modelo de M_0, \dots, M_p utilizando validación cruzada, C_p , AIC, BIC, R^2 -ajustado.

```
m_fwd = regsubsets(Salary ~ ., data = d, nvmax = 19, method = "forward")
summary(m_fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = d, nvmax = 19, method = "forward")
## 19 Variables (and intercept)
##           Forced in Forced out
## AtBat      FALSE      FALSE
## Hits      FALSE      FALSE
## HmRun      FALSE      FALSE
## Runs      FALSE      FALSE
## RBI       FALSE      FALSE
## Walks     FALSE      FALSE
## Years     FALSE      FALSE
## CAtBat     FALSE      FALSE
## CHits     FALSE      FALSE
## CHmRun    FALSE      FALSE
## CRuns     FALSE      FALSE
## CRBI      FALSE      FALSE
## CWalks    FALSE      FALSE
## LeagueN   FALSE      FALSE
## DivisionW FALSE      FALSE
## PutOuts   FALSE      FALSE
## Assists   FALSE      FALSE
## Errors    FALSE      FALSE
## NewLeagueN FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: forward
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks LeagueN DivisionW
## 1  ( 1 )  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
## 2  ( 1 )  " "  "*"  " "  " "  " "  " "  " "  " "  " "  "*"  " "  " "  " "  " "
```

```
## 3 ( 1 ) " " "*" " " " " " " " " " " " " " " " " " " " " "
## 4 ( 1 ) " " "*" " " " " " " " " " " " " " " " " " " " " "
## 5 ( 1 ) "*" "*" " " " " " " " " " " " " " " " " " " " " "
## 6 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " " " " " " "
## 7 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " " " " " " "
## 8 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " " " " " " "
## 9 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " " " " " " "
## 10 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " " " " " " "
## 11 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " " " " " " "
## 12 ( 1 ) "*" "*" " " " " "*" " " " " " " " " " " " " " " "
## 13 ( 1 ) "*" "*" " " " " "*" " " " " " " " " " " " " " " "
## 14 ( 1 ) "*" "*" "*" "*" " " " " " " " " " " " " " " " "
## 15 ( 1 ) "*" "*" "*" "*" " " " " " " " " " " " " " " " "
## 16 ( 1 ) "*" "*" "*" "*" "*" "*" " " " " " " " " " " " " " "
## 17 ( 1 ) "*" "*" "*" "*" "*" "*" " " " " " " " " " " " " " "
## 18 ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" " " " " " " " " " "
## 19 ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" " " " " " " " " " "
##
## Assists Errors NewLeagueN
## 1 ( 1 ) " " " " " "
## 2 ( 1 ) " " " " " "
## 3 ( 1 ) " " " " " "
## 4 ( 1 ) " " " " " "
## 5 ( 1 ) " " " " " "
## 6 ( 1 ) " " " " " "
## 7 ( 1 ) " " " " " "
## 8 ( 1 ) " " " " " "
## 9 ( 1 ) " " " " " "
## 10 ( 1 ) "*" " " " "
## 11 ( 1 ) "*" " " " "
## 12 ( 1 ) "*" " " " "
## 13 ( 1 ) "*" "*" " "
## 14 ( 1 ) "*" "*" " "
## 15 ( 1 ) "*" "*" " "
## 16 ( 1 ) "*" "*" " "
## 17 ( 1 ) "*" "*" "*"
## 18 ( 1 ) "*" "*" "*"
## 19 ( 1 ) "*" "*" "*"

```

```
m_fwd_summary = summary(m_fwd)
which.min(m_fwd_summary$cp)
```

```
## [1] 10
```

```
coef(m_fwd,10)
```

```
## (Intercept)      AtBat      Hits      Walks      CAtBat      CRuns      CRBI      CWalks
## 162.5354420 -2.1686501  6.9180175  5.7732246 -0.1300798  1.4082490  0.7743122 -0.83081
## DivisionW PutOuts Assists
## -112.3800575  0.2973726  0.2831680
```

3.4 Método Backward-Stepwise

Algoritmo:

1. Mp es el modelo con todos los regresores.
2. Para $k = p, \dots, 1$

- a. A partir del modelo con k regresores, M_k , estimar todos los modelos posibles con $(k-1)$ regresores.
- b. Elegir el que tenga menor RSS o mayor R^2 . Este será el modelo M_{k-1} .
3. Elegir el mejor modelo de M_0, \dots, M_p utilizando validación cruzada, C_p , AIC, BIC, R^2 ajustado.

```
m_bwd = regsubsets(Salary ~ ., data = d, nvmax = 19, method = "backward")
summary(m_bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = d, nvmax = 19, method = "backward")
## 19 Variables (and intercept)
##           Forced in Forced out
## AtBat      FALSE      FALSE
## Hits        FALSE      FALSE
## HmRun        FALSE      FALSE
## Runs         FALSE      FALSE
## RBI          FALSE      FALSE
## Walks        FALSE      FALSE
## Years        FALSE      FALSE
## CAtBat       FALSE      FALSE
## CHits        FALSE      FALSE
## CHmRun       FALSE      FALSE
## CRuns        FALSE      FALSE
## CRBI         FALSE      FALSE
## CWalks       FALSE      FALSE
## LeagueN     FALSE      FALSE
## DivisionW    FALSE      FALSE
## PutOuts      FALSE      FALSE
## Assists      FALSE      FALSE
## Errors       FALSE      FALSE
## NewLeagueN   FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: backward
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks LeagueN DivisionW
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " "
## 2 ( 1 ) " " "*" " " " " " " " " " " " " " " " "
## 3 ( 1 ) " " "*" " " " " " " " " " " " " " " " "
## 4 ( 1 ) "*" "*" " " " " " " " " " " " " " " " "
## 5 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " " "
## 6 ( 1 ) "*" "*" " " " " " " "*" " " " " " " " " "*"
## 7 ( 1 ) "*" "*" " " " " " " "*" " " " " " " "*" "*"
## 8 ( 1 ) "*" "*" " " " " " " "*" " " " " " " "*" "*"
## 9 ( 1 ) "*" "*" " " " " " " "*" " " "*" " " " "*" "*"
## 10 ( 1 ) "*" "*" " " " " " " "*" " " "*" " " " "*" "*"
## 11 ( 1 ) "*" "*" " " " " " " "*" " " "*" " " " "*" "*"
## 12 ( 1 ) "*" "*" " " "*" " " "*" " " "*" " " " "*" " "*"
## 13 ( 1 ) "*" "*" " " "*" " " "*" " " "*" " " " "*" " "*"
## 14 ( 1 ) "*" "*" "*" "*" " " "*" " " "*" " " " "*" " "*"
## 15 ( 1 ) "*" "*" "*" "*" " " "*" " " "*" " " " "*" " "*"
## 16 ( 1 ) "*" "*" "*" "*" "*" "*" " " "*" " " " "*" " "*"
## 17 ( 1 ) "*" "*" "*" "*" "*" "*" " " "*" " " " "*" " "*"
## 18 ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" " " " "*" " "*"
## 19 ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" " " " "*" " "*"
##           Assists Errors NewLeagueN
## 1 ( 1 ) " " " " " "
## 2 ( 1 ) " " " " " "
```

```
## 3 ( 1 ) " " " "
## 4 ( 1 ) " " " "
## 5 ( 1 ) " " " "
## 6 ( 1 ) " " " "
## 7 ( 1 ) " " " "
## 8 ( 1 ) " " " "
## 9 ( 1 ) " " " "
## 10 ( 1 ) "*" " " " "
## 11 ( 1 ) "*" " " " "
## 12 ( 1 ) "*" " " " "
## 13 ( 1 ) "*" "*" " "
## 14 ( 1 ) "*" "*" " "
## 15 ( 1 ) "*" "*" " "
## 16 ( 1 ) "*" "*" " "
## 17 ( 1 ) "*" "*" "*"
## 18 ( 1 ) "*" "*" "*"
## 19 ( 1 ) "*" "*" "*"

```

```
m_bwd_summary = summary(m_bwd)
which.min(m_bwd_summary$cp)

```

```
## [1] 10
coef(m_bwd,10)

```

```
## (Intercept)      AtBat      Hits      Walks      CAtBat      CRuns      CRBI      CWalks
## 162.5354420    -2.1686501    6.9180175    5.7732246    -0.1300798    1.4082490    0.7743122    -0.8308
## DivisionW      PutOuts      Assists
## -112.3800575    0.2973726    0.2831680

```

Es el mismo que antes.

3.5 Eligiendo el mejor modelo utilizando subconjuntos de validación

```
set.seed(1)
n = nrow(d)
n_train = round(0.6*n)
n_test = n - n_train

pos = 1:n
pos_train = sample(pos,n_train,replace = F) # muestreo sin reemplazamiento
pos_test = pos[-pos_train]

# dividimos los datos en training set y validation set
datos_train = d[pos_train,]
datos_test = d[pos_test,]

```

Estimamos todos los modelos posibles con los datos de entrenamiento

```
m_val = regsubsets(Salary ~ ., data = datos_train, nvmax = 19)

```

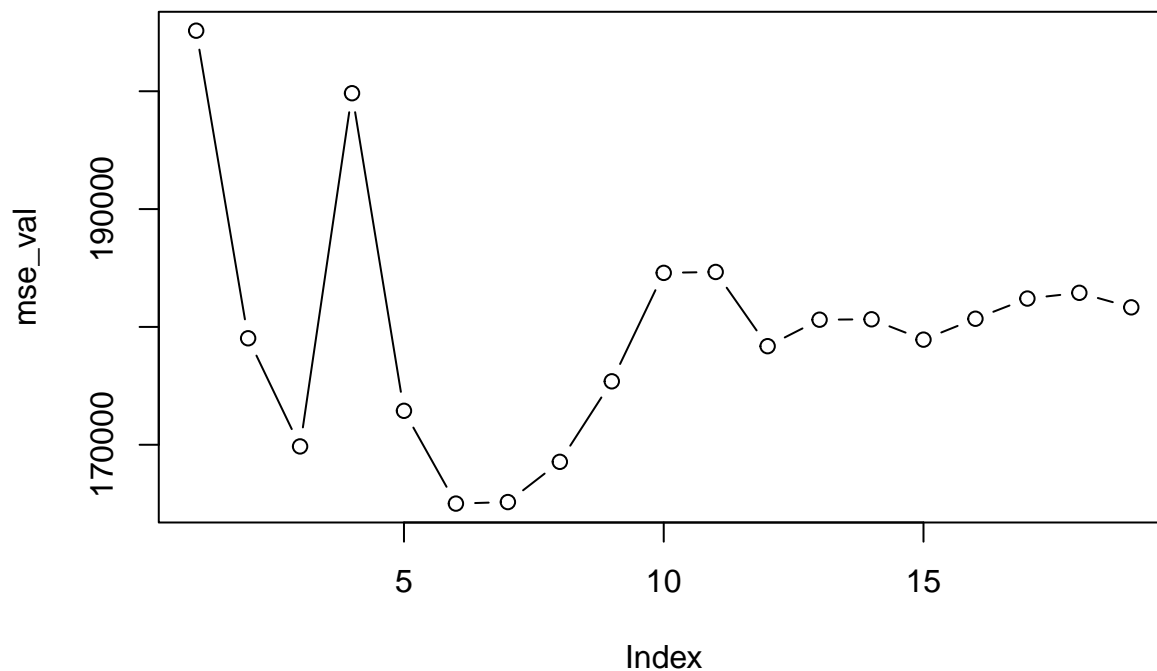
Vamos calcular el error de prediccion (MSE) de este modelo en los datos test. Como no existe la funcion predecir a partir de modelos estimados con `regsubsets()`, se ha programado en (descargar):

```
source("funciones/regsubsets_predict.R")
predict.regsubsets

```

```
## function(objeto_regsubsets, newdata, id){
##   # funcion para predecir con regsubsets
##   # objeto_regsubsets: objeto calculado con la funcion regsubsets()
##   # newdata: datos para predecir (data.frame)
##   # id: numero de variables incluidas en el modelo objeto_regsubsets
##   # -----
##
##   formu = as.formula(objeto_regsubsets$call[[2]])
##   X_mat = model.matrix(formu,newdata)
##   coefi = coef(objeto_regsubsets, id)
##   X_col = names(coefi)
##   pred = X_mat[,X_col] %*% coefi
##   return(pred)
## }
```

```
source("funciones/MSE.R")
#
mse_val = rep(0,19)
for (i in 1:19){
  pred_i = predict(m_val,datos_test,i)
  mse_val[i] = MSE(datos_test$Salary,pred_i)
}
plot(mse_val, type = "b")
```



```
which.min(mse_val)
```

```
## [1] 6
```

Para calcular los coeficientes del modelo de regresión finales, es preferible hacerlo con todos los datos:

```
m_val_final = regsubsets(Salary ~ ., data = d, nvmax = 19)
coef(m_val_final,6)
```

```
## (Intercept)      AtBat      Hits      Walks      CRBI      DivisionW      PutOuts
##  91.5117981   -1.8685892   7.6043976   3.6976468   0.6430169  -122.9515338   0.2643076
```

3.6 Eligiendo el mejor modelo utilizando Cross-Validation

Función para obtener las posiciones de train y de test:

```
source("funciones/cross_val_pos.R")
```

Datos de los folds:

```
num_folds = 10
set.seed(1)
pos = cross_val_pos(nrow(d), num_folds)
```

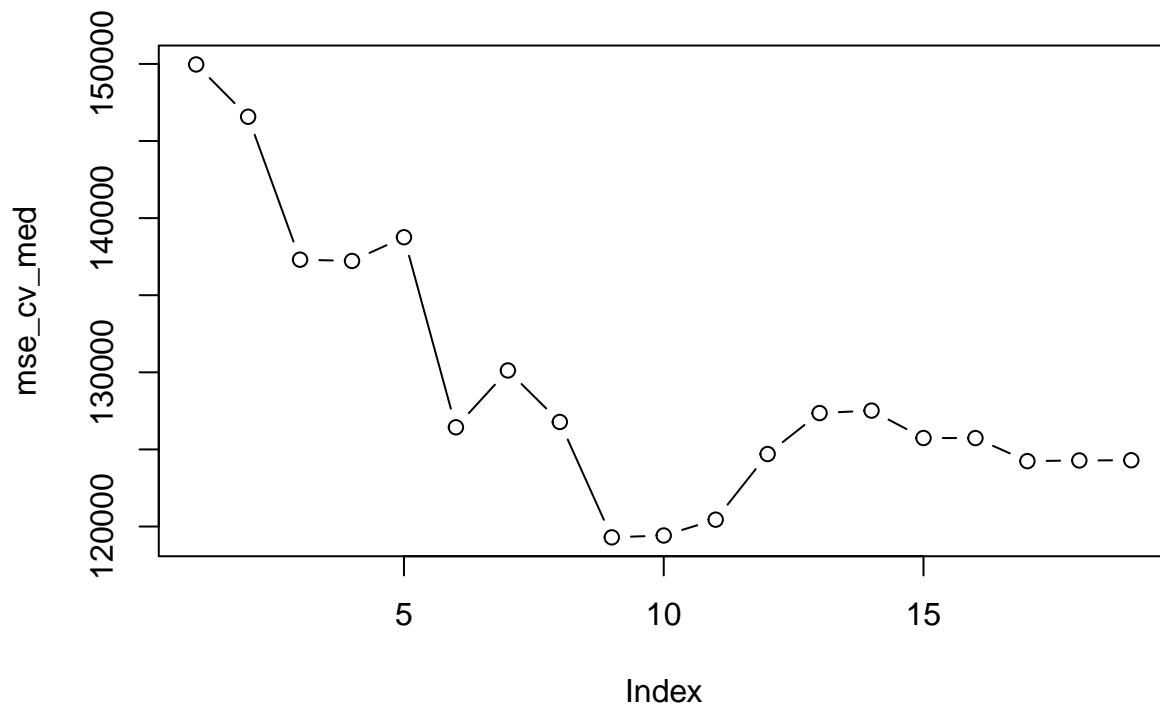
Calculamos el error cometido en cada fold por cada modelo:

```
mse_cv = matrix(0, nrow = num_folds, ncol = 19)
for (i in 1:num_folds){
  # datos de training y de validation de cada fold
  datos_train = d[pos$train[[i]],]
  datos_test = d[pos$test[[i]],]

  m_cv = regsubsets(Salary ~ ., data = datos_train, nvmax = 19)

  for (j in 1:19){
    pred = predict(m_cv, newdata = datos_test, id = j)
    mse_cv[i, j] = MSE(datos_test$Salary, pred)
  }
}
```

```
mse_cv_med = apply(mse_cv, 2, mean)
plot(mse_cv_med, type = "b")
```



El que tiene menor error es el de 9 variables. Lo aplicamos a todos los datos:

```
m_cv_final = regsubsets(Salary ~ ., data = d, nvmax = 19)
coef(m_cv_final, 9)
```

##	(Intercept)	AtBat	Hits	Walks	CAtBat	CRuns	CRBI
##	146.24960033	-1.93676754	6.65672102	5.55204413	-0.09953904	1.25067124	0.66176849
##	DivisionW	PutOuts					
##	-115.34950146	0.27773062					