ITERA

Modul 7 Praktikum
Statistika Sains Data

Linier model selection

Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

2024

A. Tujuan Praktikum

1. Mahasiswa mampu memahami penggunaan Linear Selection Model.
2. Mahasiswa dapat menerapkan model terbaik Linear Selection Model.
3. Mahasiswa dapat melakukan pengukuran terhadap Linear Selection Model.

B. Teori Dasar

Linear model selection methods — subset selection

Model linier (berganda) biasanya berbentuk

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon, \qquad (3.19)$$

Sumber : Pengantar Pembelajaran Statistika dengan Aplikasi pada R. Edisi Kedua

Ada baiknya jika kita dapat memasukkan sebanyak mungkin variabel ke dalam model kita, namun hanya jika variabel tersebut "bermakna/signifikan". Menambahkan variabel yang tidak signifikan ke dalam model berarti menambahkan gangguan pada model Anda, yang tentu saja tidak diinginkan. Mengurangi jumlah variabel suatu model juga membantu pengguna untuk memahami dan menafsirkan model. Untuk memilih model yang paling tepat, umumnya ada tiga metode: pemilihan subset, penyusutan, dan reduksi dimensi. Pada praktikum hari ini kita akan membahas tentang metode pemilihan subset, khususnya pemilihan subset terbaik dan pemilihan bertahap.

Seleksi subset terbaik dan seleksi bertahap

Pemilihan subset terbaik, dengan kata-katanya sendiri, menguji setiap kemungkinan kombinasi variabel. Mengingat jumlah p variabel/prediktor, kami akan menyesuaikan model p yang hanya berisi satu prediktor, model p(p-1)/2 yang berisi dua prediktor, dan seterusnya. Untuk setiap kelompok model n-variabel, kami memilih yang terbaik berdasarkan jumlah sisa kuadrat (RSS) atau R-kuadrat. Kemudian kita akan memiliki model p dengan masing-masing variabel 1 hingga p, dan kita membandingkan model ini menggunakan indikator seperti kesalahan prediksi validasi silang, BIC, R-squared yang disesuaikan, dll.

Algorithm 6.1 *Best subset selection*

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Sumber : Pengantar Pembelajaran Statistika dengan Aplikasi pada R. Edisi Kedua

Pemilihan subset terbaik sangat mahal secara komputasi karena harus melalui setiap model yang memungkinkan. Di sisi lain, seleksi bertahap lebih efisien dan lebih umum digunakan. Misalkan kita mempunyai 10 variabel dan ingin mencari model terbaik, kita bisa melakukannya

1. Seleksi maju: Kita memulai dengan model nol yang tidak melibatkan variabel, lalu menambahkan satu variabel setiap kali ke model hingga variabel u (u = 1, …,10) ditambahkan. *Pada setiap langkah, variabel yang memberikan peningkatan tambahan terbaik terhadap kesesuaian model ditambahkan.*
   Misalnya u sama dengan 3. Pada iterasi ini akan dibuat tiga model dengan u= 1,2,3 masing-masing. Kami memilih model terbaik di antara ketiganya menggunakan RSS atau R-squared. Ingatlah bahwa kami akan menguji semuanya dan mendapatkan 10 model terbaik pada akhirnya. Kami kemudian memilih model terbaik di antara model-model terbaik ini menggunakan kesalahan prediksi validasi silang, BIC, R-squared yang disesuaikan, dll.

Algorithm 6.2 *Forward stepwise selection*

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   (a) Consider all $p-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Choose the *best* among these $p-k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Sumber : Pengantar Pembelajaran Statistika dengan Aplikasi pada R. Edisi Kedua

2. Seleksi mundur: Kita mulai dengan model lengkap dengan 10 variabel. Kemudian kita eliminasi satu per satu variabel sampai variabel u ada dalam model, untuk u= 0, .. ,10. *Pada setiap langkah, variabel yang paling tidak berguna dihilangkan.* Sekali lagi, jika u= 3, kami memilih model terbaik yang modelnya masing-masing memiliki 10 hingga 3 variabel. Oleh karena itu, pada akhirnya kami akan memiliki 10 model terbaik. Kami kemudian memilih model terbaik di antara model-model terbaik ini menggunakan kesalahan prediksi validasi silang, BIC, R-squared yang disesuaikan, dll.

Algorithm 6.3 *Backward stepwise selection*

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Sumber : Pengantar Pembelajaran Statistika dengan Aplikasi pada R. Edisi Kedua

3. Pendekatan hibrid: Pendekatan ini menggabungkan seleksi maju dan seleksi mundur. Kami memulai dengan model nol, menambahkan satu variabel pada satu waktu dan di setiap langkah, kami juga mencari variabel yang tidak signifikan dan menghilangkannya.

Memilih model terbaik

Seperti terlihat di atas, ada dua langkah yang melibatkan perbandingan model. Langkah 2(b) dan Langkah 3. Mengapa mereka menggunakan pendekatan berbeda untuk membandingkan model? Model statistik dapat diklasifikasikan menjadi model bersarang dan tidak bersarang. " *Dua model disarangkan jika satu model berisi semua syarat dari model yang lain, dan setidaknya satu syarat tambahan. Model yang lebih besar adalah model lengkap (atau penuh), dan model yang lebih kecil adalah model tereduksi (atau dibatasi).* " Untuk model bersarang, kita dapat menggunakan RSS atau deviance (dalam kasus model nonlinier) untuk membandingkan, dan untuk model non-linear model bersarang, kita dapat menggunakan BIC, customized R-squared, dan pendekatan lainnya. Inilah sebabnya mengapa pada langkah 2(b), karena model ini merupakan hasil penambahan/penghapusan variabel dari model sebelumnya, maka model tersebut disarangkan; sedangkan ketika membandingkan model terbaik dari langkah 2(b) pada langkah 3, kita perlu menggunakan metode lain.

RSS and R-squared

Kedua metode ini hanya terkait dengan kesalahan pelatihan model karena menggunakan garis regresi (model) sebagai acuan.

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \qquad (3.16)$$

Sumber : Pengantar Pembelajaran Statistika dengan Aplikasi pada R. Edisi Kedua

Jumlah kuadrat sisa/regresi mengukur perbedaan antara setiap y yang diamati dan prediksi y yang terkait. Kita mengambil kuadrat untuk menjadikan semua perbedaan bernilai positif, terlepas dari apakah perbedaan dari nilai prediksi lebih besar atau lebih kecil dari nilai teramati.

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}} \qquad (3.17)$$

Sumber : Pengantar Pembelajaran Statistika dengan Aplikasi pada R. Edisi Kedua

dimana TSS adalah jumlah total kuadrat yang mengukur perbedaan antara setiap y yang diamati dan rata-rata prediksi y. [TSS = RSS + ESS (jumlah kesalahan kuadrat).
"TSS − RSS mengukur jumlah variabilitas respons yang dijelaskan (atau dihilangkan) dengan melakukan regresi, dan R2 mengukur proporsi variabilitas dalam Y yang dapat dijelaskan menggunakan X." R-squared terletak antara 0 dan 1, dan nilai yang mendekati 1 berarti model dapat menjelaskan sebagian besar Y dengan regresi.
Untuk mendapatkan kesalahan pengujian, kita dapat menggunakan R-squared, AIC, dan BIC yang disesuaikan untuk memperkirakan kesalahan pengujian secara "tidak langsung" dengan melakukan penyesuaian.

Adjusted R-squared
Untuk model linier, wajar saja jika semakin banyak variabel dalam suatu model akan menghasilkan R-kuadrat yang lebih besar, sehingga semakin cocok. Hal ini benar dalam situasi seperti seleksi bertahap; namun, untuk membandingkan model dengan kumpulan variabel berbeda dengan jumlah berbeda, membandingkan model dengan R-kuadrat saja sudah menyesatkan. Oleh karena itu, kami menggunakan R-squared yang disesuaikan yang menambahkan jumlah observasi dan variabel ke dalam rumus.

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}. \qquad (6.4)$$

Sumber : Pengantar Pembelajaran Statistika dengan Aplikasi pada R. Edisi Kedua. d: jumlah variabel. n: jumlah pengamatan

Dengan menggunakan R-squared yang disesuaikan, " *setelah semua variabel yang benar dimasukkan ke dalam model, menambahkan variabel noise tambahan hanya akan menyebabkan penurunan RSS yang sangat kecil"* . Hal ini karena menambahkan variabel noise yang tidak akan meningkatkan RSS akan menurunkan nominator rumus 6.4, dan karenanya menurunkan R-kuadrat yang disesuaikan secara keseluruhan.

Akaike's Information Criterion (AIC), Bayesian Information Criterion (BIC)

Untuk model non-linier, AIC dan BIC biasanya digunakan untuk membandingkan model non-nested.

C. Latihan Praktikum

Pada praktikum kali ini kita akan menggunakan Best Subset selection untuk Hitters data. Yang mana kita melakukan prediksi gaji seorang pemain baseball dengan variasi statistik yang mempertimbangkan performa dari tahun lalu.

```
library(ISLR)
fix(Hitters)
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"
## [19] "Salary"    "NewLeague"
```

Berikut dimensi hitters data

```
dim(Hitters)
```

```
## [1] 322  20
```

Apakah terdapat missing data pada atribut salary. kita akan memeriksanya sebagai berikut:

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

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

Kemudian kita menghapus dataset yang missing

```
Hitters=na.omit(Hitters)
dim(Hitters)
```

```
## [1] 263  20
```

Periksa kembali apakah masih terdapat missing data

```
sum(is.na(Hitters))
```

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

Nah, selanjutnya kita akan menerapkan regsubsets() function (yang merupakan bagian dari leaps library). fungsi ini dapat menunnjukkan jumlah prediktor yang diberikan, dimana best merupakan kuantifikasi menggunakan RSS () . Sintaks yang diterapkan juga sama dengan `lm()`. lalu ditampilkan melalui fungsi `summary()`.

```
library(leaps)
regfit.full=regsubsets(Salary~.,  Hitters)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., Hitters)
## 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
## 1  ( 1 ) " "   " "  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 2  ( 1 ) " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 3  ( 1 ) " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 4  ( 1 ) " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 5  ( 1 ) "*"   "*"  " "   " "  " " " " " "   " "   " "   " "    " "   " "   "*"
## 6  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   " "   " "    " "   " "   "*"
## 7  ( 1 ) " "   "*"  " "   " "  " " " " "*"   " "   "*"   "*"    "*"   " "   " "
## 8  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   " "   " "    "*"   "*"   " "
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
```

Tanda bintang menunjukkan bahwa variabel yang diberikan termasuk dalam model yang sesuai. Misalnya, keluarannya menunjukkan bahwa model dua variabel hanya mengandung `Hits` dan `CRBI`. secara default,

`regsubset()` hanya menampilkan 8 variabel model terbaik. tetapi opsi `nvmax` dapat digunakan untuk menampilkan kembali sebanyak variabel yang diinginkan. Berikut contohnya ketika kita menggunakan 19 variabel model

```
regfit.full=regsubsets(Salary~., data=Hitters, nvmax=19)
reg.summary=summary(regfit.full)
```

fungsi `summary()` juga mengembalikan R^2, RSS, dan pengaturan R^2, C_p, dan BIC. kita akan mencoba menjelaskan model ini dengan memilih best dari semua model.

```
names(reg.summary)
```

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

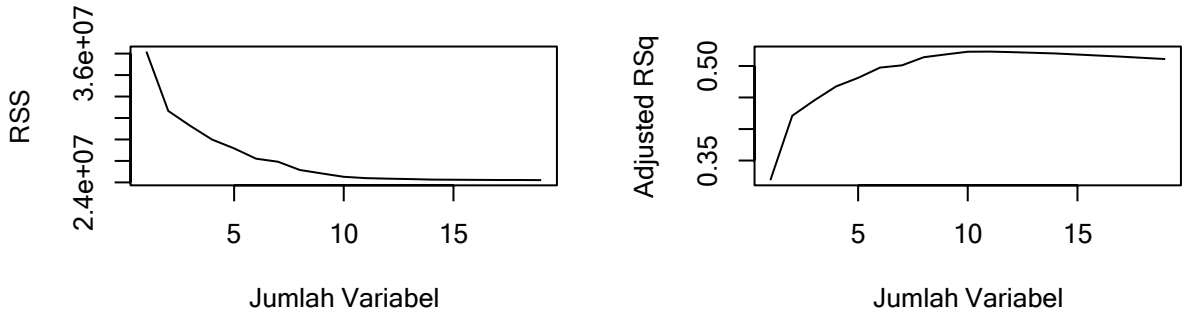
Misalnya, kita melihat bahwa statistik R^2 meningkat dari 32%, ketika hanya satu variabel yang dimasukkan ke dalam model, menjadi hampir 55%, ketika semua variabel dimasukkan. Seperti yang diharapkan, statistik R^2 meningkat secara monoton karena lebih banyak variabel dimasukkan.

```
reg.summary$rsq
```

```
##  [1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227
##  [8] 0.5285569 0.5346124 0.5404950 0.5426153 0.5436302 0.5444570 0.5452164
## [15] 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159
```

Melakukan plotting RSS, adjusted R^2, C_p, dan BIC untuk semua model dapat membantu kita memutuskan model mana yang akan kita gunakan. pada contoh di bawah ini `type="l"` memberikan plot point dengan line.

```
par(mfrow=c(2,2))
plot(reg.summary$rss, xlab="Jumlah Variabel", ylab="RSS", type = "l")
plot(reg.summary$adjr2, xlab="Jumlah Variabel", ylab="Adjusted RSq", type = "l")
```

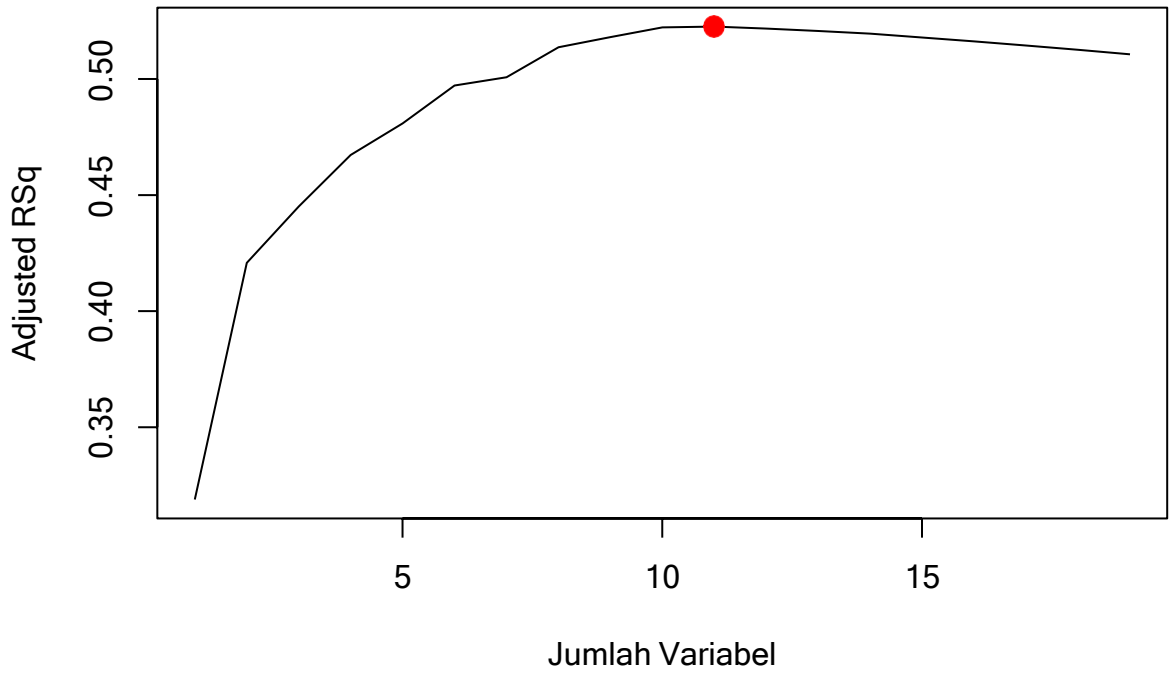


Perintah points() bekerja seperti perintah plot(), kecuali bahwa points() menempatkan titik pada plot yang telah dibuat. Fungsi which.max() dapat digunakan untuk mengidentifikasi lokasi titik maksimum vektor. Kita sekarang akan memplot titik merah untuk menunjukkan model dengan statistik R^2 terbesar yang disesuaikan.

```
which.max(reg.summary$adjr2)
```

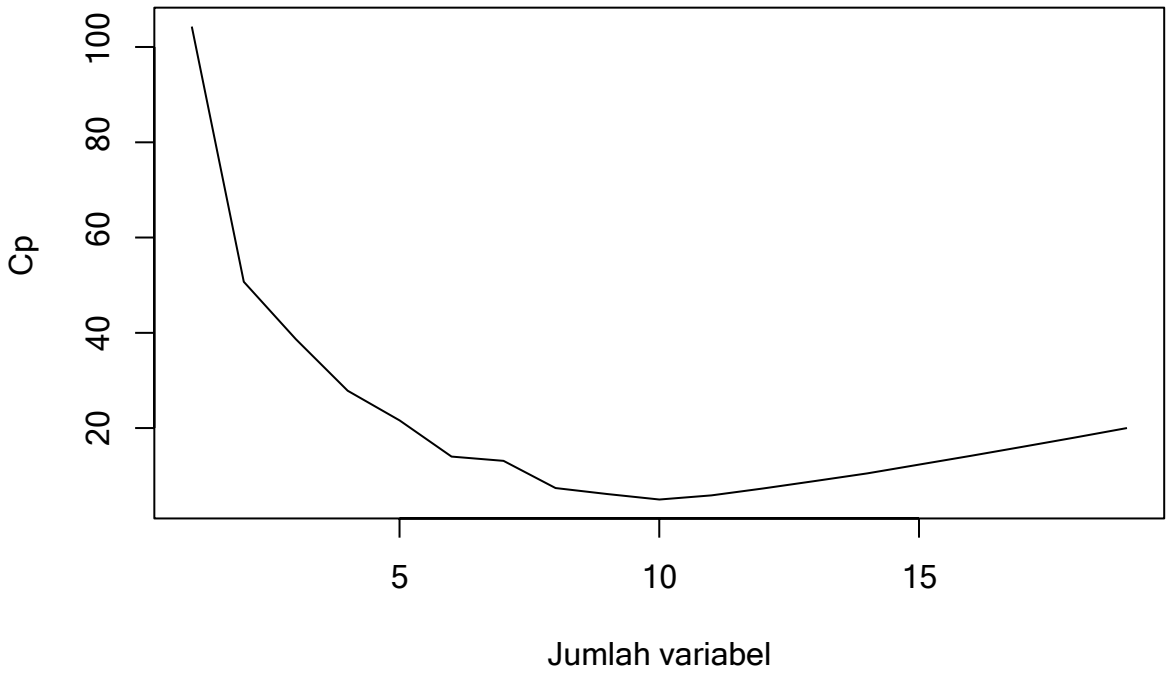
```
## [1] 11
```

```
plot(reg.summary$adjr2, xlab="Jumlah Variabel", ylab="Adjusted RSq", type = "l")
points(11,reg.summary$adjr2[11], col="red", cex=2, pch=20)
```



Kita juga dapat memplotkan C_p dan BIC statistik,dan indikasi model dengan statistik paling kecil melalui fungsi `which.min()`.

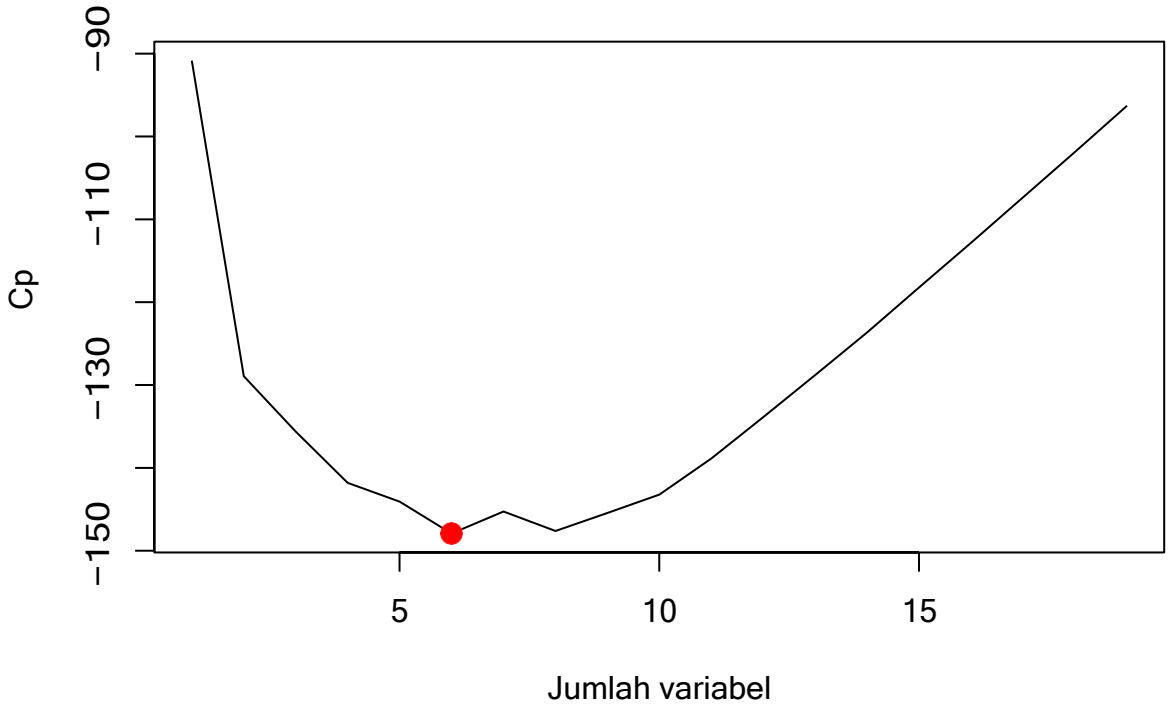
```
plot(reg.summary$cp, xlab ="Jumlah variabel",ylab="Cp", type="l")
```




```
which.min(reg.summary$cp)
```

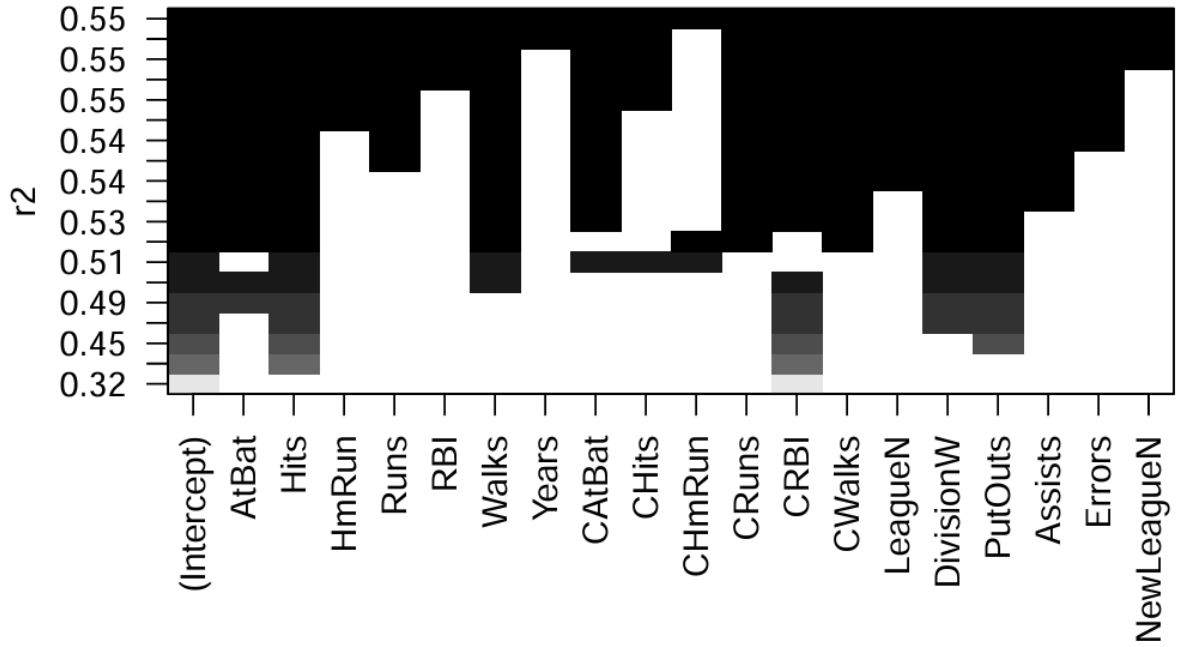
```
## [1] 10
```

```
plot(reg.summary$bic, xlab ="Jumlah variabel",ylab="Cp", type="l")
points(6, reg.summary$bic[6],col="red", cex=2, pch=20)
```

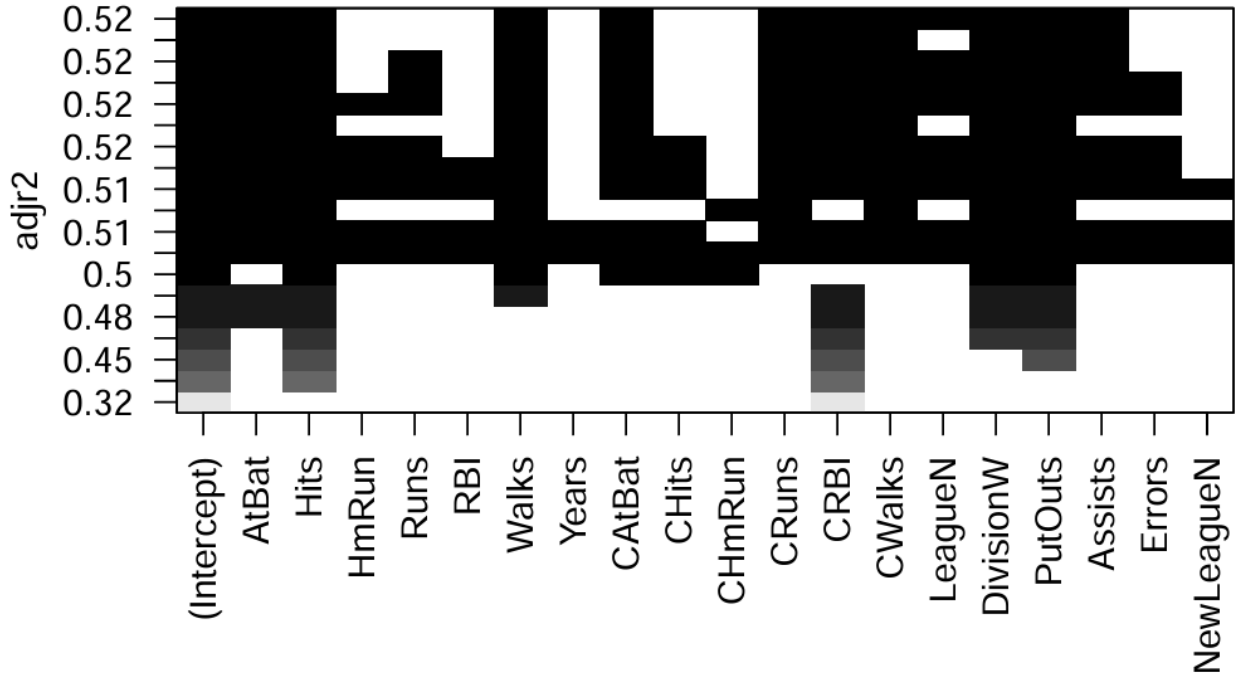


fungsi `regsubsets()` merupakan built-in dari package `plot()`. yang mana dapat digunakan untuk memilih variabel untuk model terbaik dengan memberikan jumlah prediktor, urutan peringkat pada BIC, C_p, adjusted R^2 atau AIC. untuk memeriksa lanjut dapat menggunakan perintah berikut: `?plot.regsubsets`

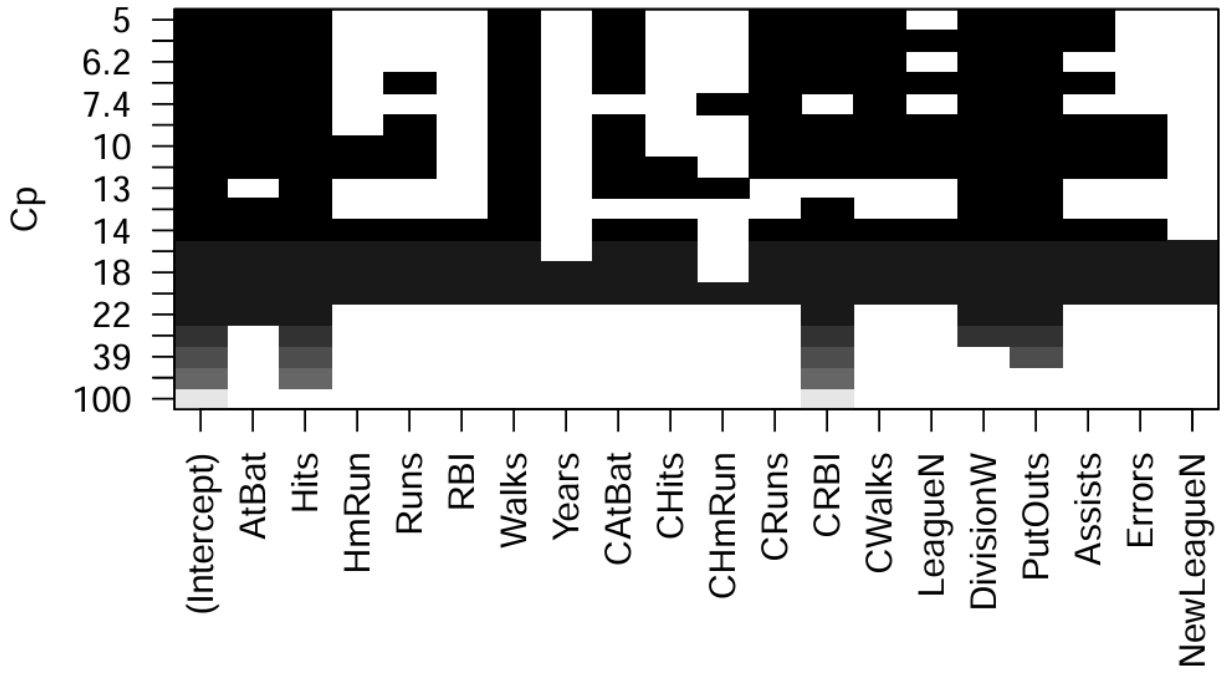
```
plot(regfit.full, scale="r2")
```



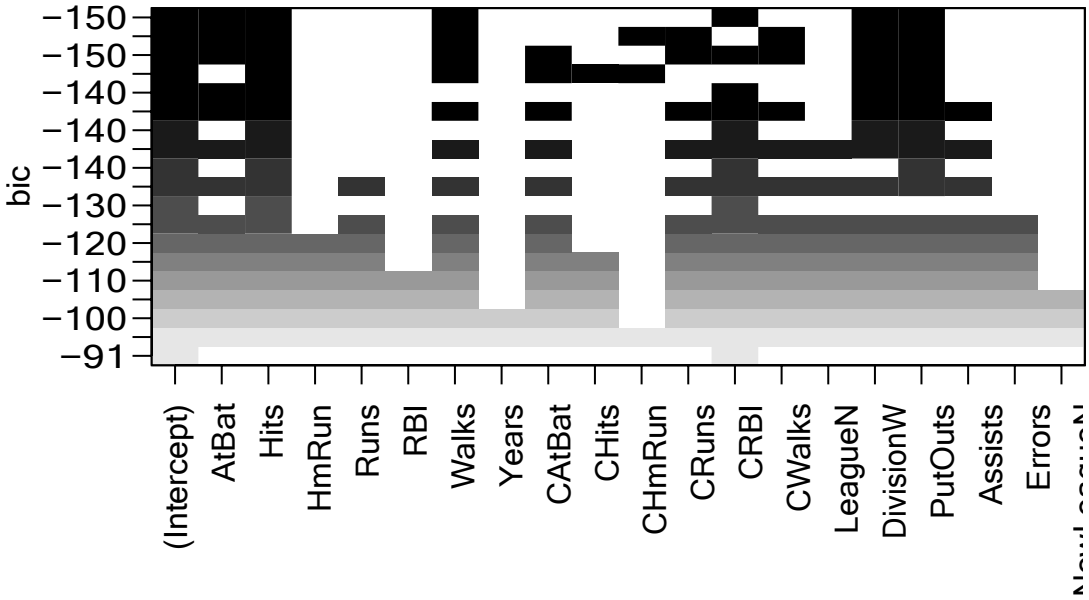

```
plot(regfit.full, scale="adjr2")
```




```
plot(regfit.full, scale="Cp")
```




```
plot(regfit.full, scale="bic")
```



Baris atas setiap plot berisi kotak hitam untuk setiap variabel yang dipilih sesuai dengan model optimal yang terkait dengan statistik tersebut. Misalnya, kita melihat bahwa beberapa model berbagi BIC mendekati 150. Namun, model dengan BIC terendah adalah model enam variabel yang hanya berisi AtBat, Hits, Walks, CRBI, DivisionW, dan PutOuts. Kita dapat menggunakan fungsi coef() untuk melihat perkiraan koefisien yang terkait dengan model ini.

```
coef(regfit.full,6)
```

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

Forward and Backward Stepwise Selection

kita bisa menggunakan fungsi regsubset() untuk forward stepwise atau backward stepwise selection, dengan menggunakan argumen `method="forward"` atau `method="backward"`.

```
regfit.fwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="forward")
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, 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
## 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 ) "*"   "*"  "*"   "*"  "*" "*" "*"   "*"    "*"   "*"    "*"   "*"
##          CWalks LeagueN DivisionW PutOuts 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 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```
regfit.bwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="backward")
summary(regfit.bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, 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
## 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 ) "*"   "*"  "*"   "*"  "*" "*" "*"   "*"   "*"   "*"    "*"   "*"
##           CWalks LeagueN DivisionW PutOuts 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 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

Misalnya, kita melihat bahwa menggunakan forward stepwise selection, model satu variabel terbaik hanya berisi `CRBI`, dan model dua variabel terbaik juga mencakup `Hits`. Untuk data ini, model satu variabel hingga enam variabel terbaik masing-masing identik untuk subset terbaik dan seleksi maju. Namun, model tujuh variabel terbaik yang diidentifikasi dengan pemilihan forward stepwise, pemilihan backward stepwise, dan pemilihan subset terbaik berbeda.

```
coef(regfit.full,7)
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits       CHmRun
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073    1.4420538
##    DivisionW      PutOuts
## -129.9866432    0.2366813
```

```
coef(regfit.full,7)
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits       CHmRun
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073    1.4420538
##    DivisionW      PutOuts
## -129.9866432    0.2366813
```

```
coef(regfit.bwd,7)
```

```
##  (Intercept)         AtBat         Hits        Walks        CRuns       CWalks
##  105.6487488   -1.9762838    6.7574914    6.0558691    1.1293095   -0.7163346
##    DivisionW      PutOuts
## -116.1692169    0.3028847
```

Memilih model dengan menggunakan Pendekatan Validation Set dan Cross Validation

Pendekatan ini bertujuan agar menghasilkan perkiraan yang akurat dari kesalahan pengujian. kita harus menggunakan hanya pengamatan pelatihan untuk melakukan semua aspek pemasangan model—termasuk pemilihan variabel. Oleh karena itu, penentuan model mana dari ukuran tertentu yang terbaik harus dibuat hanya dengan menggunakan pengamatan pelatihan. Jika kumpulan data lengkap digunakan untuk melakukan langkah pemilihan subset terbaik, kesalahan validation set dan kesalahan cross validation yang kita peroleh tidak akan menjadi estimasi akurat dari kesalahan pengujian. Untuk menggunakan pendekatan set validasi, kita mulai dengan membagi pengamatan menjadi set pelatihan dan set tes.

```
set.seed(1)
train=sample(c(TRUE,FALSE), nrow(Hitters), rep=TRUE)
test=(!train)
```

Selanjutnya kita menerapkan `regsubsets()` pada training set untuk memerintahkan best subset selection.

```
regfit.best=regsubsets(Salary~., data=Hitters[train,],nvmax = 19)
```

kemudian untuk validation set

```
test.mat=model.matrix(Salary~.,data=Hitters[test,])
```

Fungsi `model.matrix()` digunakan di banyak paket regresi untuk membangun model matriks "X" dari data. Sekarang kita menjalankan loop, dan untuk setiap ukuran i, matrix() mengekstrak koefisien dari regfit.best untuk model terbaik dari ukuran itu, mengalikannya ke dalam kolom yang sesuai dari matriks model uji untuk membentuk prediksi, dan menghitung uji MSE.

```
val.errors=rep(NA,19)
for(i in 1:19){
  coefi=coef(regfit.best,id=i)
  pred=test.mat[,names(coefi)]%*%coefi
  val.errors[i]=mean((Hitters$Salary[test]-pred)^2)
}
```

kita akan mendapatkan hasil model terbaik yang terdiri dari 10 variabel.

```
val.errors
```

```
##  [1] 164377.3 144405.5 152175.7 145198.4 137902.1 139175.7 126849.0 136191.4
##  [9] 132889.6 135434.9 136963.3 140694.9 140690.9 141951.2 141508.2 142164.4
## [17] 141767.4 142339.6 142238.2
```

```
which.min(val.errors)
```

```
## [1] 7
```

```
coef(regfit.best,10)
```

```
##  (Intercept)        AtBat         Hits        HmRun        Walks       CAtBat
##   71.8074075   -1.5038124    5.9130470  -11.5241809    8.4349759   -0.1654850
##        CRuns         CRBI       CWalks    DivisionW      PutOuts
##    1.7064330    0.7903694   -0.9107515 -109.5616997    0.2426078
```

tidak ada metode predict() untuk regsubsets(). Karena kita akan menggunakan fungsi ini lagi, kita dapat menangkap langkah-langkah kita di atas dan menulis metode prediksi kita sendiri.

```
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}
```

Terakhir kita akan menjalankan program untuk best subset selection pada full dataset, dan pilih 10 variabel model terbaik. Penting bagi kita untuk menggunakan kumpulan data lengkap untuk mendapatkan estimasi koefisien yang lebih akurat. Perhatikan bahwa kita melakukan pemilihan subset terbaik pada kumpulan data lengkap dan memilih model sepuluh variabel terbaik, daripada hanya menggunakan variabel yang diperoleh dari kumpulan pelatihan, karena model sepuluh variabel terbaik pada kumpulan data lengkap mungkin berbeda dari yang sesuai model pada set pelatihan.

```
regfit.best=regsubsets(Salary~.,data=Hitters,nvmax=19)
coef(regfit.best,10)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  162.5354420   -2.1686501    6.9180175    5.7732246   -0.1300798    1.4082490
##         CRBI       CWalks    DivisionW      PutOuts      Assists
##    0.7743122   -0.8308264 -112.3800575    0.2973726    0.2831680
```

Kita sekarang mencoba memilih di antara model dengan ukuran berbeda menggunakan cross validation. Pendekatan ini agak sulit, karena kita harus melakukan pemilihan subset terbaik dalam setiap set pelatihan k. Pertama, kita membuat vektor yang mengalokasikan setiap pengamatan ke salah satu dari k = 10 folds, dan kita membuat matriks tempat kita akan menyimpan hasilnya.

```
k=10
set.seed(1)
folds=sample(1:k,nrow(Hitters),replace=TRUE)
cv.errors=matrix(NA,k,19, dimnames=list(NULL,paste(1:19)))
```

kemudian dilakukan loop for untuk cross validation. Pada lipatan ke-j, elemen lipatan yang sama dengan j ada di set pengujian, dan sisanya ada di set pelatihan. Kita membuat prediksi untuk setiap ukuran model (menggunakan metode predict() baru), menghitung error pengujian pada subset yang sesuai, dan menyimpannya di slot yang sesuai di matriks `cv.errors`.

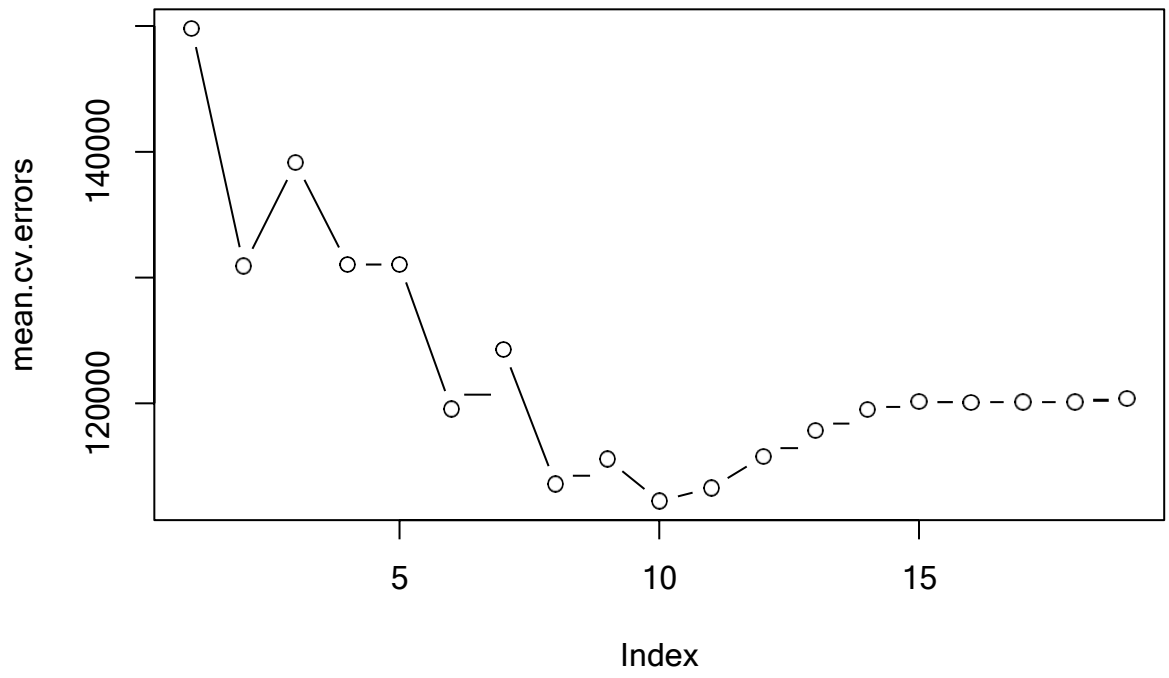
```
for(j in 1:k){
  best.fit=regsubsets(Salary~.,data=Hitters[folds!=j,],nvmax=19)
  for(i in 1:19){
    pred=predict(best.fit,Hitters[folds==j,],id=i)
    cv.errors[j,i]=mean((Hitters$Salary[folds==j]-pred)^2)
  }
}
```

hasilnya akan memberi kita matriks 10 × 19, di mana elemen (i, j) sesuai dengan uji MSE untuk lipatan validasi silang ke-i untuk model variabel-j terbaik. Kita menggunakan fungsi apply() untuk menghitung rata-rata kolom matriks ini untuk mendapatkan vektor elemen ke-j adalah cross validation error untuk model variabel-j.

```
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 149821.1 130922.0 139127.0 131028.8 131050.2 119538.6 124286.1 113580.0
##        9       10       11       12       13       14       15       16
## 115556.5 112216.7 113251.2 115755.9 117820.8 119481.2 120121.6 120074.3
##       17       18       19
## 120084.8 120085.8 120403.5
```

```
par(mfrow=c(1,1))
plot(mean.cv.errors,type="b")
```



kita melihat bahwa cross validation memilih 11 variabel model. kita selanjutnya memilih best subset selection pada full dataset untuk menghitung 11 variabel model.

```
reg.best=regsubsets(Salary~.,data=Hitters,nvmax=19)
coef(reg.best,11)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  135.7512195   -2.1277482    6.9236994    5.6202755   -0.1389914    1.4553310
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##    0.7852528   -0.8228559   43.1116152 -111.1460252    0.2894087    0.2688277
```

Ridge Selection dan Lasso

Kita akan menggunakan package glmnet untuk melakukan regresi ridge dan laso. Fungsi utama dalam paket ini adalah glmnet(), yang dapat digunakan glmnet() agar sesuai dengan model regresi ridge, model laso, dan lainnya. Fungsi ini memiliki sintaks yang sedikit berbeda dari fungsi model-fitting lainnya yang telah kita jumpai sejauh ini. Secara khusus, kita harus melewatkan matriks x dan juga vektor y, dan kita tidak menggunakan sintaks y~x.

```
x=model.matrix(Salary~.,Hitters)[,-1]
y=Hitters$Salary
```

Fungsi model.matrix() sangat berguna untuk membuat x; tidak hanya menghasilkan matriks yang sesuai dengan 19 prediktor tetapi juga secara otomatis mengubah variabel kualitatif apa pun menjadi variabel dummy. Properti yang terakhir penting karena glmnet() hanya dapat mengambil input numerik dan kuantitatif.

Ridge Regression

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
```

Secara default fungsi glmnet() melakukan regresi ridge untuk rentang nilai lambda yang dipilih secara otomatis. Namun, di sini kita telah memilih untuk mengimplementasikan fungsi pada kisi nilai mulai dari lambda = 10^10 hingga lambda = 10^{-2}, yang pada dasarnya mencakup berbagai skenario dari model nol yang hanya berisi intersep, hingga kuadrat terkecil yang sesuai. Seperti yang akan kita lihat, kita juga dapat menghitung kecocokan model untuk nilai lambda tertentu yang bukan merupakan salah satu dari nilai grid asli. Perhatikan bahwa secara default, fungsi glmnet() membakukan variabel sehingga berada pada skala yang sama. Untuk mematikan pengaturan default ini, gunakan argumen standardize=FALSE.

Terkait dengan setiap nilai lambda adalah vektor koefisien regresi ridge, disimpan dalam matriks yang dapat diakses oleh coef(). Dalam hal ini, adalah 20×100. matriks, dengan 20 baris (satu untuk setiap prediktor, ditambah intersep) dan 100 kolom (satu untuk setiap nilai lambda).

```
dim(coef(ridge.mod))
```

```
## [1]  20 100
```

Kita berharap perkiraan koefisien menjadi jauh lebih kecil, dalam hal l_2 norm, ketika nilai lambda yang besar digunakan, dibandingkan dengan ketika nilai lambda yang kecil digunakan. koefisien ketika lambda = 11.498, bersama dengan l_2 norm:

```
ridge.mod$lambda[50]
```

```
## [1] 11497.57
```

```
coef(ridge.mod)[,50]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs 
## 407.356050200   0.036957182   0.138180344   0.524629976   0.230701523 
##           RBI         Walks         Years        CAtBat         CHits 
##   0.239841459   0.289618741   1.107702929   0.003131815   0.011653637 
##        CHmRun         CRuns          CRBI        CWalks       LeagueN 
##   0.087545670   0.023379882   0.024138320   0.025015421   0.085028114 
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN 
##  -6.215440973   0.016482577   0.002612988  -0.020502690   0.301433531
```

```
sqrt(sum(coef(ridge.mod)[-1,50]^2))
```

```
## [1] 6.360612
```

berikut koefisien ketika lambda =705, bersama dengan l_2 norm nya. Perhatikan l_2 norm yang jauh lebih besar dari koefisien yang terkait dengan nilai lambda yang lebih kecil ini.

```
ridge.mod$lambda[60]
```

```
## [1] 705.4802
```

```
coef(ridge.mod)[,60]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs          RBI 
## 54.32519950  0.11211115  0.65622409  1.17980910  0.93769713  0.84718546
```

```
##        Walks        Years       CAtBat        CHits       CHmRun        CRuns
##   1.31987948   2.59640425   0.01083413   0.04674557   0.33777318   0.09355528
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##   0.09780402   0.07189612  13.68370191 -54.65877750   0.11852289   0.01606037
##       Errors   NewLeagueN
##  -0.70358655   8.61181213
```

```
sqrt(sum(coef(ridge.mod)[-1,60]^2))
```

```
## [1] 57.11001
```

kita dapat menggunakan fungsi predict() untuk beberapa tujuan. seperti, kita dapat menghitung ridge regression coefficient untuk nilai baru dari lambda, katakanlah 50:

```
predict(ridge.mod,s=50,type="coefficients")[1:20,]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  4.876610e+01 -3.580999e-01  1.969359e+00 -1.278248e+00  1.145892e+00
##           RBI         Walks         Years        CAtBat         CHits
##  8.038292e-01  2.716186e+00 -6.218319e+00  5.447837e-03  1.064895e-01
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##  6.244860e-01  2.214985e-01  2.186914e-01 -1.500245e-01  4.592589e+01
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -1.182011e+02  2.502322e-01  1.215665e-01 -3.278600e+00 -9.496680e+00
```

Selanjutnya kita membagi data menjadi training set dan test set untuk mengestimasi error dari ridge regression dan lasso.

```
set.seed(1)
train=sample(1:nrow(x),nrow(x)/2)
test=(-train)
y.test=y[test]
```

kemudian kita lakukan fitting untuk ridge model pada training set dan evaluasi dengan MSE pada test set, menggunakan lambda = 4. gunakan fungsi predict(), ganti argumen type="coefficients" dengan newx argumen

```
ridge.mod=glmnet(x[train,], y[train], alpha=0, lambda=grid, thresh = 1e-12)
ridge.pred=predict(ridge.mod, s=4,newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
## [1] 142199.2
```

MSE yang dihasilkan adalah 142199,2 . Dapat juga dilakukan perhitungan MSE dengan sintaks berikut

```
mean((mean(y[train])-y.test)^2)
```

```
## [1] 224669.9
```

Kita juga bisa mendapatkan hasil yang sama dengan memasang model regresi ridge dengan nilai lambda yang sangat besar. Perhatikan bahwa 1e10 berarti 10^10.

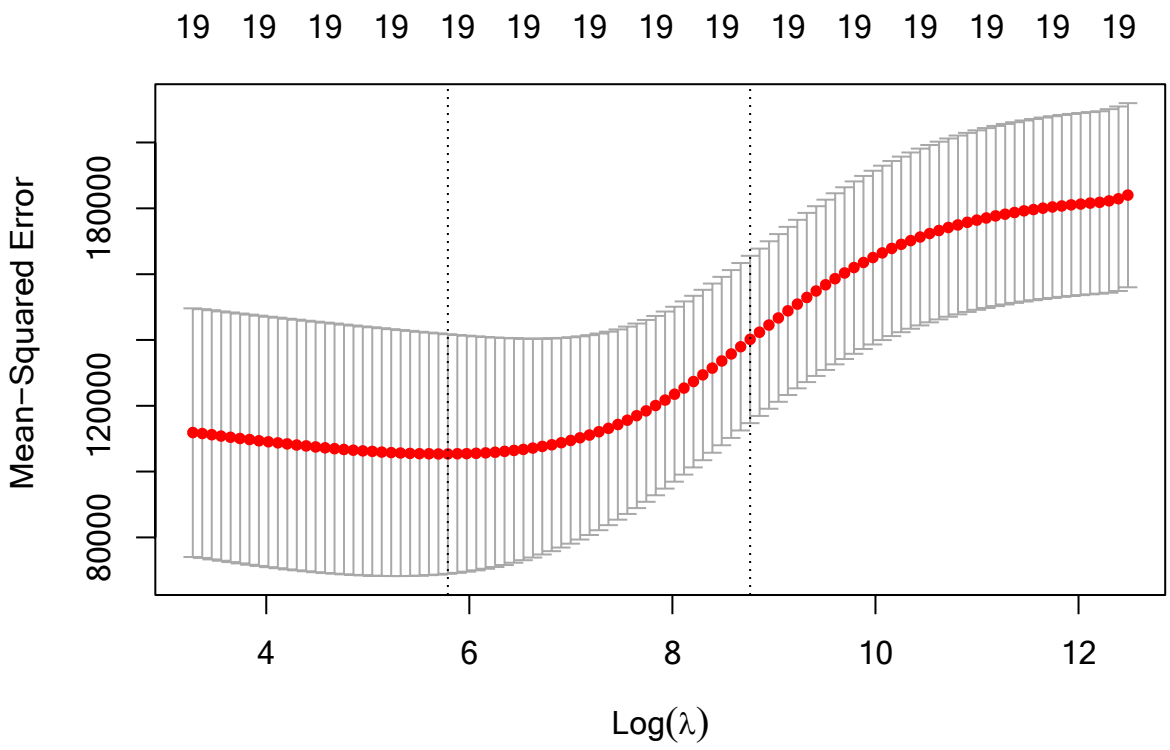
```
ridge.pred=predict(ridge.mod, s=1e10, newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
## [1] 224669.8
```

Jadi fitting model regresi ridge dengan lambda = 4 mengarah ke tes MSE yang jauh lebih rendah daripada fitting model hanya dengan intersep. Kita sekarang memeriksa apakah ada manfaat untuk melakukan regresi ridge dengan lambda = 4 daripada hanya melakukan regresi kuadrat terkecil. Ingatlah bahwa kuadrat terkecil hanyalah regresi ridge dengan lambda = 0^5.

Menggunakan CV (Cross validation). Secara default, fungsi cv.glmnet() melakukan cross validation dalam 10 fold, meskipun hal ini dapat diubah menggunakan lipatan argumen. Perhatikan bahwa kita menetapkan seed random terlebih dahulu sehingga hasilnya akan direproduksi, karena pilihan cross validation fold adalah acak.

```
set.seed (1)
cv.out =cv.glmnet (x[train ,],y[train],alpha =0)
plot(cv.out)
```




```
bestlam =cv.out$lambda.min
bestlam
```

```
## [1] 326.0828
```

```
ridge.pred=predict (ridge.mod ,s=bestlam ,newx=x[test,])
mean(( ridge.pred -y.test)^2)
```

```
## [1] 139856.6
```

Terakhir kita melakukan fitting dengan model ini pada full dataset, gunakan lambda yang di pilih oleh cross validation:

```
out=glmnet(x,y,alpha =0)
predict(out  ,type="coefficients",s=bestlam)[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs          RBI
## 15.44383120   0.07715547   0.85911582   0.60103106   1.06369007   0.87936105
##        Walks        Years       CAtBat        CHits       CHmRun        CRuns
##  1.62444617   1.35254778   0.01134999   0.05746654   0.40680157   0.11456224
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
```

```
##    0.12116504    0.05299202   22.09143197  -79.04032656    0.16619903    0.02941950
##        Errors    NewLeagueN
##   -1.36092945    9.12487765
```

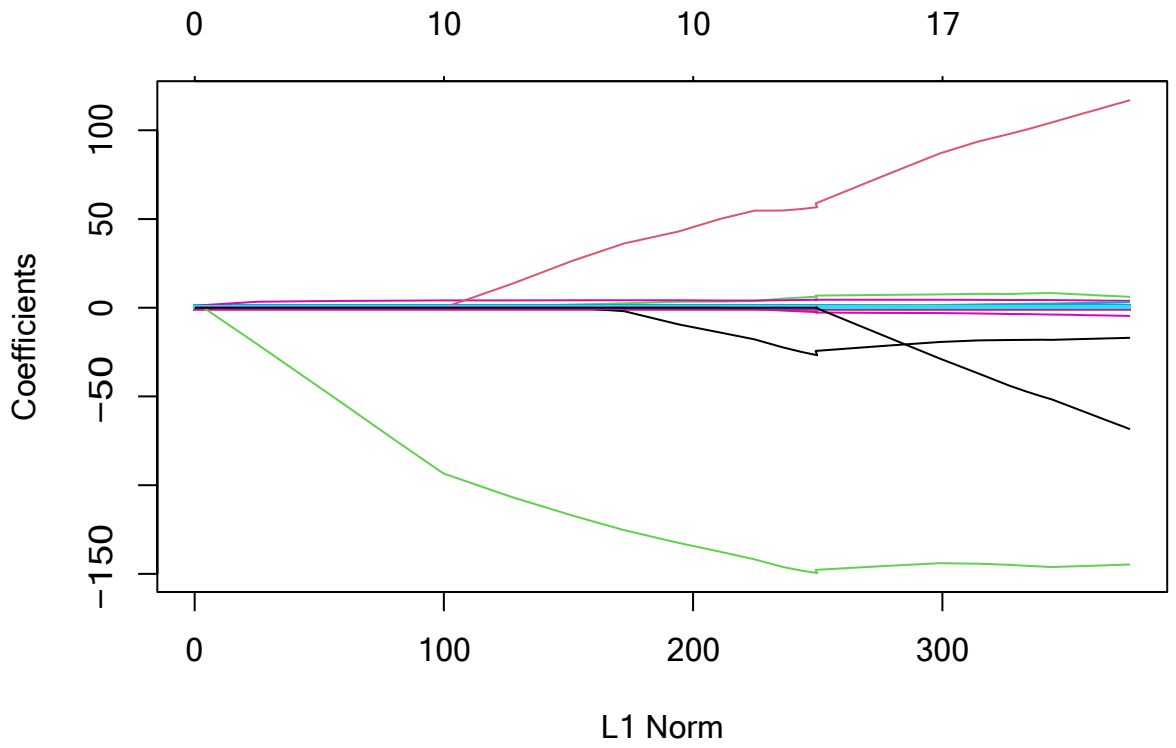
Seperti yang diharapkan, tidak ada koefisien yang nol—regresi ridge tidak melakukan pemilihan variabel!

Lasso

Kita melihat bahwa regresi ridge dengan pilihan lambda dapat mengungguli kuadrat terkecil serta model nol pada kumpulan data Hitters. lalu, apakah laso dapat menghasilkan model yang lebih akurat atau lebih dapat ditafsirkan daripada regresi ridge?. Agar sesuai dengan model laso, kita sekali lagi menggunakan fungsi `glmnet()`; namun, kali ini kita menggunakan argumen alpha=1. Selain perubahan itu, kita melanjutkan seperti yang kita lakukan saat fitting model ridge.

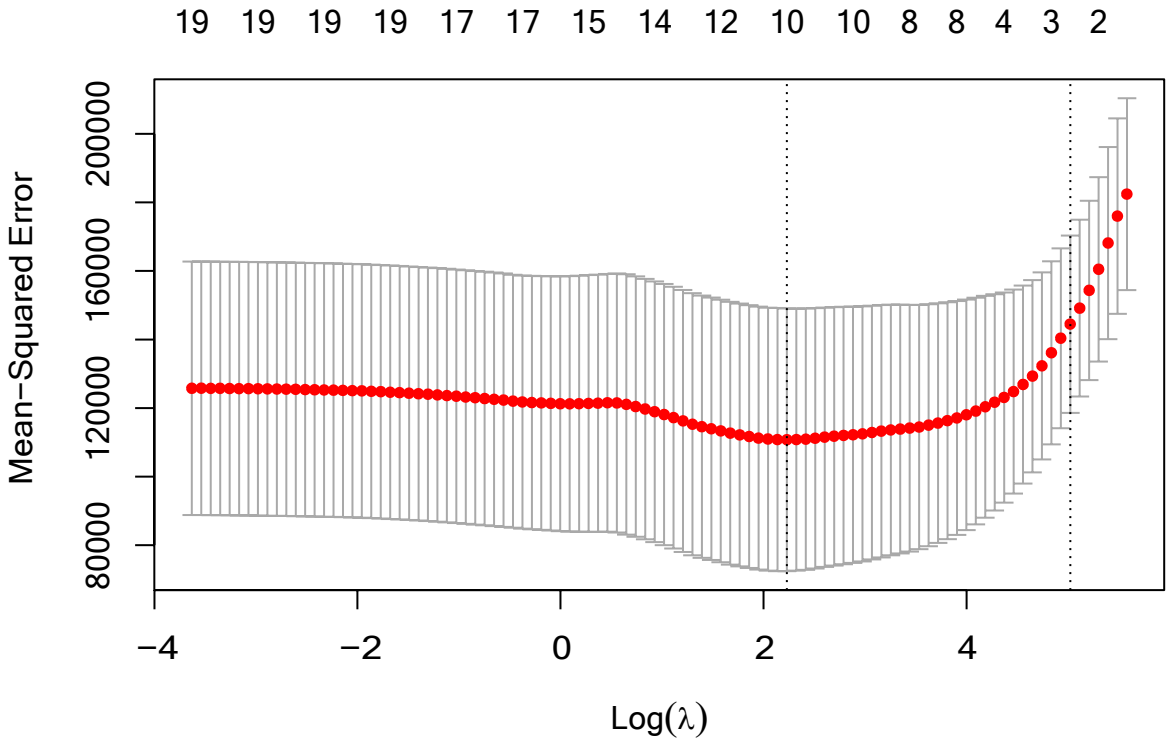
```
lasso.mod=glmnet(x[train ,],y[train],alpha =1, lambda =grid)
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):##
collapsing to unique "x" values
```



Kita dapat melihat dari plot koefisien yang bergantung pada pilihan parameter penalaan, beberapa koefisien akan sama persis dengan nol. Kemudian lakukan cross validation dan menghitung kesalahan pengujian.

```
set.seed (1)
cv.out =cv.glmnet (x[train ,],y[train],alpha =1)
plot(cv.out)
```




```
bestlam =cv.out$lambda.min
lasso.pred=predict(lasso.mod ,s=bestlam ,newx=x[test,])
mean(( lasso.pred -y.test)^2)
```

```
## [1] 143673.6
```

secara substansial lebih rendah daripada uji MSE dari model nol dan kuadrat terkecil, dan sangat mirip dengan uji MSE regresi ridge dengan lambda dipilih dengan cross validation.

Namun, laso memiliki keunggulan substansial dibandingkan regresi ridge karena perkiraan koefisien yang dihasilkan parse. Di sini kita melihat bahwa 12 dari 19 estimasi koefisien persis nol. Jadi model laso dengan lambda yang dipilih dengan cross validation hanya berisi tujuh variabel.

```
out=glmnet (x,y,alpha=1, lambda=grid)
lasso.coef=predict (out ,type ="coefficients",s=bestlam)[1:20,]
lasso.coef
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##    1.27479059   -0.05497143    2.18034583    0.00000000    0.00000000
##           RBI         Walks         Years        CAtBat         CHits
##    0.00000000    2.29192406   -0.33806109    0.00000000    0.00000000
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##    0.02825013    0.21628385    0.41712537    0.00000000   20.28615023
##      DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -116.16755870    0.23752385    0.00000000   -0.85629148    0.00000000
```