統計學習 作業五

106070038 科管院學士班 杜葳葳

8.

(a) 使用 set.seed()以確保重覆執行時，每次 random 產生的數值相同。

用 rnorm()產生 100 個 X 和 100 個 epsilon

```
> set.seed(10)
> X <- rnorm(100)
> X
  [1]  0.01874617 -0.18425254 -1.37133055 -0.59916772  0.29454513  0.38979430 -1.20807618
  [8] -0.36367602 -1.62667268 -0.25647839  1.10177950  0.75578151 -0.23823356  0.98744470
 [15]  0.74139013  0.08934727 -0.95494386 -0.19515038  0.92552126  0.48297852 -0.59631064
 [22] -2.18528684 -0.67486594 -2.11906119 -1.26519802 -0.37366156 -0.68755543 -0.87215883
 [29] -0.10176101 -0.25378053 -1.85374045 -0.07794607  0.96856634  0.18492596 -1.37994358
 [36] -1.43551436  0.36208723 -1.75908675 -0.32454401 -0.65156299  1.08655140 -0.76254488
 [43] -0.82866254  0.83447390 -0.96765199 -0.02881534  0.23252515 -0.30120868 -0.67761458
 [50]  0.65522764 -0.40063755 -0.33455657  1.36795395  2.13776710  0.50581926  0.78634238
 [57] -0.90221194  0.53289699 -0.64589425  0.29098749 -1.23759447 -0.45617628 -0.83032265
 [64]  0.34011564  1.06637640  1.21612584  0.73569066 -0.48120862  0.56274476 -1.24631971
 [71]  0.38092221 -1.43042725 -1.04844550 -0.21850355 -1.48993624  1.17270628 -1.47982702
 [78] -0.43038782 -1.05163864  1.52258634  0.59282805 -0.22266151  0.71289428  0.71660083
 [85]  0.44024186  0.15883062  0.65976414  2.22051966 -1.18394507 -0.07395583 -0.41635467
 [92] -0.19148234  0.06954478  1.15534832  0.59495735 -1.41964511 -1.60667725  0.89292590
 [99]  0.14816796  1.22702839
> set.seed(10)
> epsilon <- rnorm(100)
> epsilon
  [1]  0.01874617 -0.18425254 -1.37133055 -0.59916772  0.29454513  0.38979430 -1.20807618
  [8] -0.36367602 -1.62667268 -0.25647839  1.10177950  0.75578151 -0.23823356  0.98744470
 [15]  0.74139013  0.08934727 -0.95494386 -0.19515038  0.92552126  0.48297852 -0.59631064
 [22] -2.18528684 -0.67486594 -2.11906119 -1.26519802 -0.37366156 -0.68755543 -0.87215883
 [29] -0.10176101 -0.25378053 -1.85374045 -0.07794607  0.96856634  0.18492596 -1.37994358
 [36] -1.43551436  0.36208723 -1.75908675 -0.32454401 -0.65156299  1.08655140 -0.76254488
 [43] -0.82866254  0.83447390 -0.96765199 -0.02881534  0.23252515 -0.30120868 -0.67761458
 [50]  0.65522764 -0.40063755 -0.33455657  1.36795395  2.13776710  0.50581926  0.78634238
 [57] -0.90221194  0.53289699 -0.64589425  0.29098749 -1.23759447 -0.45617628 -0.83032265
 [64]  0.34011564  1.06637640  1.21612584  0.73569066 -0.48120862  0.56274476 -1.24631971
 [71]  0.38092221 -1.43042725 -1.04844550 -0.21850355 -1.48993624  1.17270628 -1.47982702
 [78] -0.43038782 -1.05163864  1.52258634  0.59282805 -0.22266151  0.71289428  0.71660083
 [85]  0.44024186  0.15883062  0.65976414  2.22051966 -1.18394507 -0.07395583 -0.41635467
 [92] -0.19148234  0.06954478  1.15534832  0.59495735 -1.41964511 -1.60667725  0.89292590
 [99]  0.14816796  1.22702839
```

(b) 將 beta0 設為 4、beta1 設為 3、beta2 設為 2、beta3 設為 1

，帶入 y = beta0 + beta1*x + beta2*x^2 + beta3*x^3 + epsilon，生成 100 個 Y

```
> Y <- b0+b1*X+b2*X^2+b3*X^3+epsilon
> Y
  [1]  4.0756941  3.3246326 -0.3030795  2.1062307  5.3772480  5.9222816  0.3234670  2.7617165
  [9] -1.5188427  3.0887773 12.1724242  8.5972441  3.1470552 10.8626779  8.4723912  4.3740682
 [17]  1.1332298  3.2881338 10.2080562  6.5111142  2.1138903 -5.6259807  1.9040607 -5.0108796
 [25]  0.1154246  2.7324280  1.8702134  1.3692695  3.6126128  3.0973424 -2.9123623  3.6998933
 [33] 10.6591391  4.8144231 -0.3390354 -0.5788211  5.7580355 -2.2908684  2.8782978  1.9662058
 [41] 11.9901695  1.6693693  1.4896857  9.3116721  1.0960315  3.8863754  5.0508087  2.9492910
 [49]  1.8967302  7.7608615  2.6541642  2.8481837 15.7742614 31.4608635  6.6643988  8.8682607
 [57]  1.2847368  6.8508780  1.9813280  5.3579364  0.2173530  2.4965597  1.4851267  5.6311640
 [65] 11.7524619 13.6210314  8.4234299  2.4268595  7.0625533  0.1854224  5.8691648 -0.5562936
 [73]  0.8522028  3.2110412 -0.8274493 13.0540578 -0.7801875  2.5691938  0.8422798 18.2566487
 [81]  7.2825489  3.1974711  8.2303195  8.2614234  6.2339178  4.6897837  7.7968219 33.6922269
 [89]  0.4081051  3.7147111  2.6091081  3.3003808  4.2881884 12.8332460  7.2983775 -0.5089377
 [97] -1.4113810  9.8782816  4.6398321 13.7667232
```

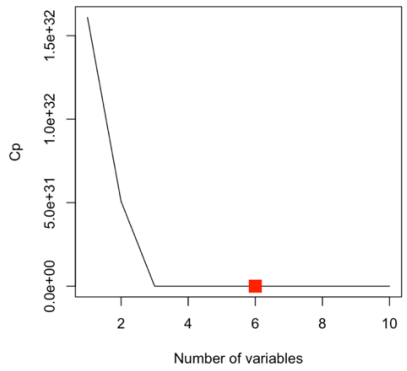
(c) 先將前兩小題產生的 X 和 Y 放入 dataframe 以方便後面做計算，用 regsubsets()以產生最佳的 subset selection，結果如下圖方框，可知分別選擇 1 到 10 個變數時應該考慮哪些 predictors，舉例來說：若選擇四個變數、則最好的模式應該是考慮 X、X^2、X^3、X^9。

(因為^用在公式中會被視為公式運算符，故使用 **I( )**以將之用做算術運算符)

```
Subset selection object
Call: regsubsets.formula(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) +
    I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10), data = data,
    nvmax = 10)
10 Variables  (and intercept)
        Forced in Forced out
x           FALSE      FALSE
I(x^2)      FALSE      FALSE
I(x^3)      FALSE      FALSE
I(x^4)      FALSE      FALSE
I(x^5)      FALSE      FALSE
I(x^6)      FALSE      FALSE
I(x^7)      FALSE      FALSE
I(x^8)      FALSE      FALSE
I(x^9)      FALSE      FALSE
I(x^10)     FALSE      FALSE
1 subsets of each size up to 10
Selection Algorithm: exhaustive
          x   I(x^2) I(x^3) I(x^4) I(x^5) I(x^6) I(x^7) I(x^8) I(x^9) I(x^10)
1  ( 1 )  "*" " "    " "    " "    " "    " "    " "    " "    " "    " "
2  ( 1 )  "*" "*"    " "    " "    " "    " "    " "    " "    " "    " "
3  ( 1 )  "*" "*"    "*"    " "    " "    " "    " "    " "    " "    " "
4  ( 1 )  "*" "*"    "*"    " "    " "    " "    " "    " "    "*"    " "
5  ( 1 )  "*" "*"    "*"    " "    " "    " "    "*"    " "    "*"    " "
6  ( 1 )  "*" "*"    "*"    " "    "*"    " "    "*"    " "    "*"    " "
7  ( 1 )  "*" "*"    "*"    "*"    "*"    " "    "*"    " "    "*"    " "
8  ( 1 )  "*" "*"    "*"    " "    "*"    " "    "*"    "*"    "*"    "*"
9  ( 1 )  "*" "*"    "*"    " "    "*"    "*"    "*"    "*"    "*"    "*"
10  ( 1 ) "*" "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"
```

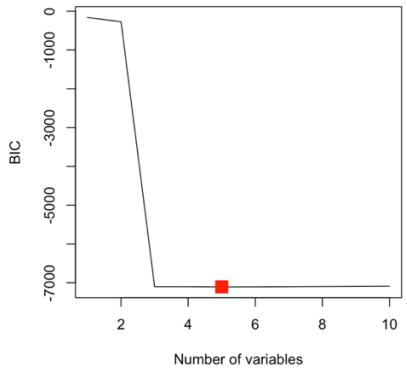
接著依序用 Cp、BIC、Adjr^2 來選擇最佳的模式

- **Cp**：最佳模式為 **6 個變數**時，下面左圖為選擇不同數量的變數分別對應的 Cp，由圖可知 Cp 最小的為選擇 6 個變數的模式(如標示的實心方點所示) 。左圖為最佳模式時各變數的係數。



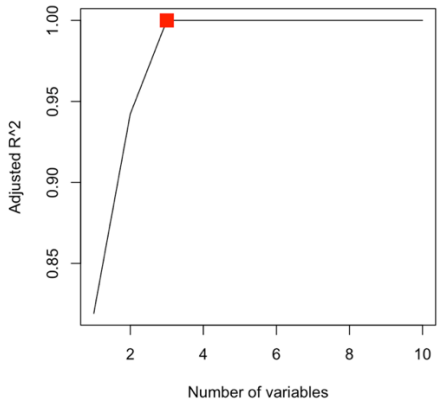

```
> coef(regfit, which.min(regsum$cp))
  (Intercept)             x        I(x^2)        I(x^3)        I(x^5)        I(x^7)        I(x^9)
 4.000000e+00  4.000000e+00  2.000000e+00  1.000000e+00 -8.881784e-16 -9.020562e-17  4.163336e-17
```

- **BIC**：最佳模式為 **5 個變數**時，下面左圖為選擇不同數量的變數分別對應的 BIC，由圖可知 BIC 最小的為選擇 5 個變數的模式(如標示的實心方點所示) 。左圖為最佳模式時各變數的係數。




```
> coef(regfit, which.min(regsum$bic))
  (Intercept)             x        I(x^2)        I(x^3)        I(x^7)        I(x^9)
 4.000000e+00  4.000000e+00  2.000000e+00  1.000000e+00  9.714451e-17 -1.387779e-17
```

- **Adj R^2**：最佳模式為 **3 個變數**時，下面左圖為選擇不同數量的變數分別對應的 Adjr^2，由圖可知 Adjr^2 最大的為選擇 3 個變數的模式(如標示的實心方點所示)。左圖為最佳模式時各變數的係數。




```
> coef(regfit, which.max(regsum$adjr2))
(Intercept)           x      I(x^2)      I(x^3)
          4           4           2           1
```

由上面三個圖可觀察到，使用 Cp、BIC、Adj R^2 挑選最佳模式時，結果可能不同，故實際在選模時應根據需求的不同，選擇適合的準則、來選模式。

以下為三者分別的特性：

- Adj R^2：越**高**越好，調整後(對模型複雜度加入懲罰)解釋變異量佔總變異量的百分比
- Cp：越**低**越好，以控制參數數量
- BIC：越**低**越好，考慮了樣本數量，樣本數量過多時，可有效防止模型精度過高造成的模型複雜度過高

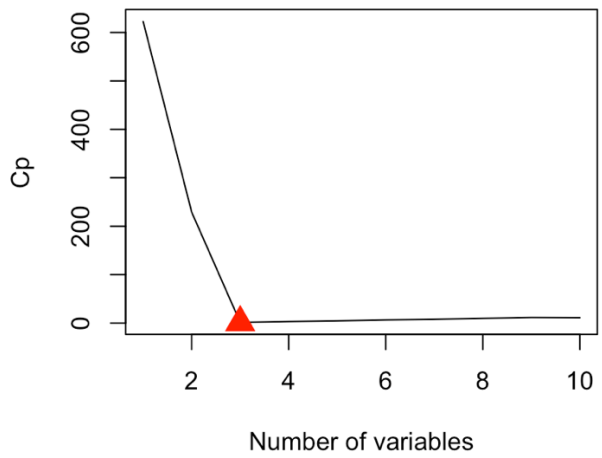
(d) 以 Forward、Backward selection 重覆(c)

Forward Selection：在一個空的迴歸中，**逐一添加變數，不顯著的自變項不挑選**，結果與(c)不同

```
Subset selection object
Call: regsubsets.formula(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) +
    I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10), data = data,
    nvmax = 10, method = "forward")
10 Variables  (and intercept)
        Forced in Forced out
x           FALSE      FALSE
I(x^2)      FALSE      FALSE
I(x^3)      FALSE      FALSE
I(x^4)      FALSE      FALSE
I(x^5)      FALSE      FALSE
I(x^6)      FALSE      FALSE
I(x^7)      FALSE      FALSE
I(x^8)      FALSE      FALSE
I(x^9)      FALSE      FALSE
I(x^10)     FALSE      FALSE
1 subsets of each size up to 10
Selection Algorithm: forward
          x   I(x^2) I(x^3) I(x^4) I(x^5) I(x^6) I(x^7) I(x^8) I(x^9) I(x^10)
1  ( 1 )  " " " "    "*"    " "    " "    " "    " "    " "    " "    " "
2  ( 1 )  " " "*"    "*"    " "    " "    " "    " "    " "    " "    " "
3  ( 1 )  "*" "*"    "*"    " "    " "    " "    " "    " "    " "    " "
4  ( 1 )  "*" "*"    "*"    "*"    " "    " "    " "    " "    " "    " "
5  ( 1 )  "*" "*"    "*"    "*"    " "    " "    " "    "*"    " "    " "
6  ( 1 )  "*" "*"    "*"    "*"    " "    " "    " "    "*"    " "    "*"
7  ( 1 )  "*" "*"    "*"    "*"    " "    "*"    " "    "*"    " "    "*"
8  ( 1 )  "*" "*"    "*"    "*"    " "    "*"    "*"    "*"    " "    "*"
9  ( 1 )  "*" "*"    "*"    "*"    "*"    "*"    "*"    "*"    " "    "*"
10  ( 1 ) "*" "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"
```

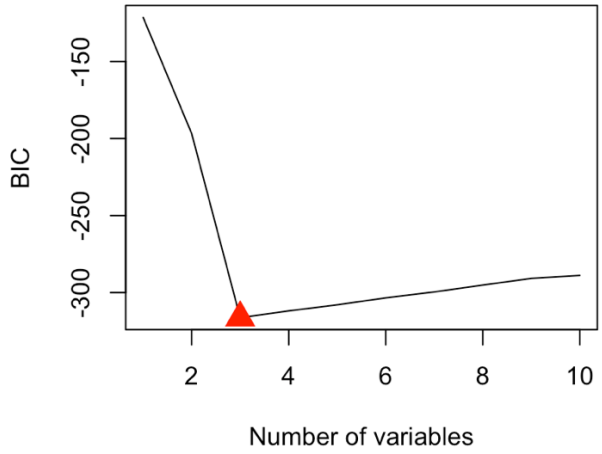
- Cp：最佳模式為 **3 個變數**時，下面左圖為選擇不同數量的變數分別對應的 Cp，由圖可知 Cp 最小的為選擇 3 個變數的模式(如標示的實心三角形所示) 。左圖為最佳模式時各變數的係數。

  Y = 3.928974 + 2.884212*X + 1.963622*X^2 + 1.021113*X^3



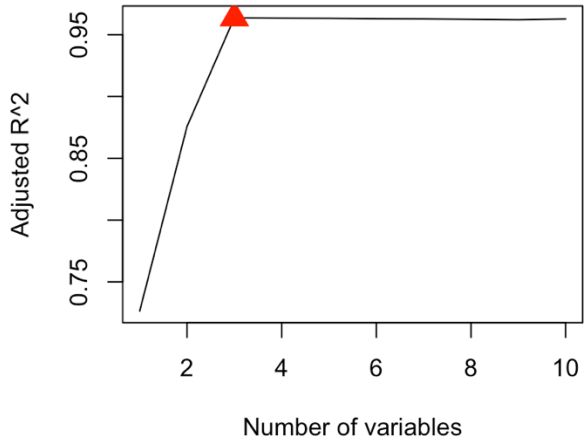

```
> coef(regfit_fwd, which.min(regsum.fwd$cp))
(Intercept)           x      I(x^2)      I(x^3)
   3.928974    2.884212    1.963622    1.021113
```

- BIC：最佳模式為 **3 個變數**時，下面左圖為選擇不同數量的變數分別對應的 BIC，由圖可知 BIC 最小的為選擇 3 個變數的模式(如標示的實心三角形所示) 。左圖為最佳模式時各變數的係數。
  Y = 3.928974 + 2.884212*X + 1.963622*X^2 + 1.021113*X^3




```
> coef(regfit_fwd, which.min(regsum.fwd$bic))
(Intercept)           x      I(x^2)      I(x^3)
   3.928974    2.884212    1.963622    1.021113
```

- Adj R^2：最佳模式為 **3 個變數**時，下面左圖為選擇不同數量的變數分別對應的 Adj R^2，由圖可知 Adj R^2 最小的為選擇 3 個變數的模式(如標示的實心三角形所示) 。左圖為最佳模式時各變數的係數。
  Y = 3.928974 + 2.884212*X + 1.963622*X^2 + 1.021113*X^3

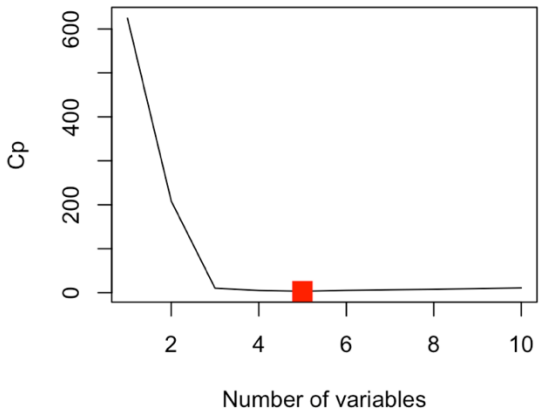



```
> coef(regfit_fwd, which.max(regsum.fwd$adjr2))
(Intercept)           x      I(x^2)      I(x^3)
   3.928974    2.884212    1.963622    1.021113
```

Backward selection：向後選取和向前選取相反。首先是將所有的自變項都選至模型中，再**一步一步篩選最不顯著的變數**，一個一個從模型中挑選出來，直到所有在模型中的自變項都是顯著的，結果與(c)和 forward 皆不同

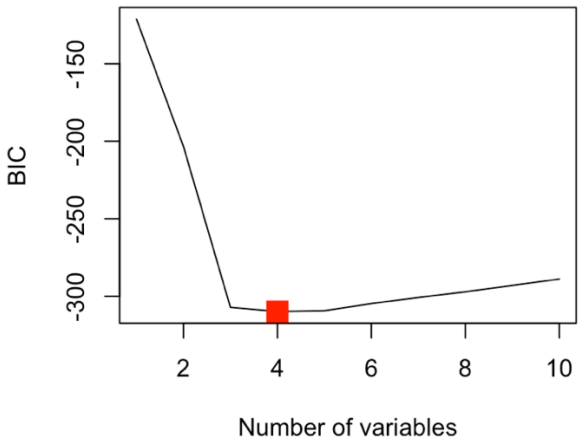
```
Subset selection object
Call: regsubsets.formula(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) +
    I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10), data = data,
    nvmax = 10, method = "backward")
10 Variables  (and intercept)
        Forced in Forced out
x           FALSE      FALSE
I(x^2)      FALSE      FALSE
I(x^3)      FALSE      FALSE
I(x^4)      FALSE      FALSE
I(x^5)      FALSE      FALSE
I(x^6)      FALSE      FALSE
I(x^7)      FALSE      FALSE
I(x^8)      FALSE      FALSE
I(x^9)      FALSE      FALSE
I(x^10)     FALSE      FALSE
1 subsets of each size up to 10
Selection Algorithm: backward
          x   I(x^2) I(x^3) I(x^4) I(x^5) I(x^6) I(x^7) I(x^8) I(x^9) I(x^10)
1  ( 1 )  "*" " "    " "    " "    " "    " "    " "    " "    " "    " "
2  ( 1 )  "*" "*"    " "    " "    " "    " "    " "    " "    " "    " "
3  ( 1 )  "*" "*"    " "    " "    "*"    " "    " "    " "    " "    " "
4  ( 1 )  "*" "*"    " "    " "    "*"    " "    "*"    " "    " "    " "
5  ( 1 )  "*" "*"    " "    " "    "*"    " "    "*"    " "    "*"    " "
6  ( 1 )  "*" "*"    " "    " "    "*"    " "    "*"    "*"    "*"    " "
7  ( 1 )  "*" "*"    " "    " "    "*"    "*"    "*"    "*"    "*"    " "
8  ( 1 )  "*" "*"    "*"    " "    "*"    "*"    "*"    "*"    "*"    " "
9  ( 1 )  "*" "*"    "*"    " "    "*"    "*"    "*"    "*"    "*"    "*"
10  ( 1 ) "*" "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"
```

- Cp：最佳模式為 **5 個變數**時，下面左圖為選擇不同數量的變數分別對應的 Cp，由圖可知 Cp 最小的為選擇 5 個變數的模式(如標示的正方形所示) 。左圖為最佳模式時各變數的係數。Y = 3.95409012 + 3.12434155*X + 1.93068856*X^2 + 1.04218301*X^5 – 0.36785066*X^7 + 0.04036764*X^9



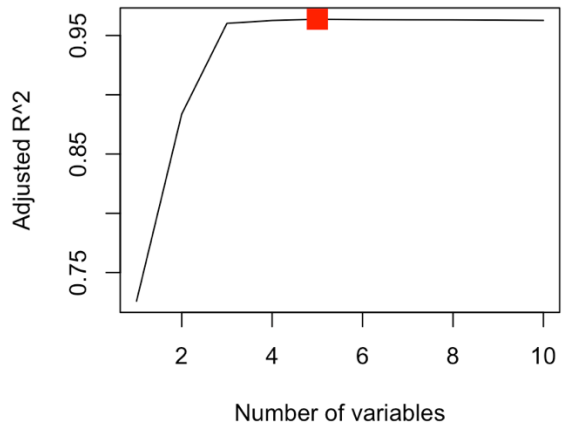

```
> coef(regfit_bwd, which.min(regsum.bwd$cp))
(Intercept)           x      I(x^2)      I(x^5)      I(x^7)      I(x^9)
 3.95409012  3.12434155  1.93068856  1.04218301 -0.36785066  0.04036764
```

- BIC：最佳模式為 **4 個變數**時，下面左圖為選擇不同數量的變數分別對應的 BIC，由圖可知 BIC 最小的為選擇 4 個變數的模式(如標示的正方形所示) 。左圖為最佳模式時各變數的係數。Y = 3.92543184 + 3.42357607*X + 1.97564883*X^2 + 0.46966736*X^5 – 0.05868386*X^7




```
> coef(regfit_bwd, which.min(regsum.bwd$bic))
(Intercept)           x      I(x^2)      I(x^5)      I(x^7)
 3.92543184  3.42357607  1.97564883  0.46966736 -0.05868386
```

- Adj R^2：最佳模式為 **5 個變數**時，下面左圖為選擇不同數量的變數分別對應的 Adj R^2，由圖可知 Adj R^2 最小的為選擇 5 個變數的模式(如標示的正方形所示) 。左圖為最佳模式時各變數的係數。Y = 3.95409012 + 3.12434155*X + 1.93068856*X^2 + 1.04218301*X^5 – 0.36785066*X^7 + 0.04036764*X^9



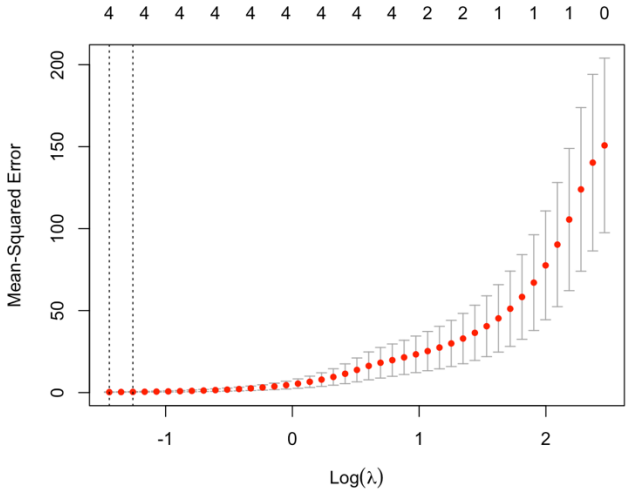

```
> coef(regfit_bwd, which.max(regsum.bwd$adjr2))
(Intercept)           x      I(x^2)      I(x^5)      I(x^7)      I(x^9)
 3.95409012  3.12434155  1.93068856  1.04218301 -0.36785066  0.04036764
```

由以上可得知，使用 exhaustive、forward、backward 的最佳模式皆不同。

(e)

Lasso 的 alpha=1

下圖為交叉驗證的散點圖，橫軸為 logλ，縱軸為均方誤差，每個點的標準偏差上界和下界如圖，頂部字數表示非零係數的個數，兩條垂直的虛線即為交叉驗證後選定的 λ。




```
> cv$lambda.min
[1] 0.2358726
> cv$lambda.1se
[1] 0.2841094
```

- lambda.min 是指交叉驗證中使得均方誤差最小的那個值 λ
- lambda.1se 為離最小均方誤差一倍標準差的 λ 值

回歸模型的係數：

```
                     1
(Intercept) 1.212561547
x           2.808334597
I(x^2)      2.696012706
I(x^3)      3.940388336
I(x^4)      0.006666267
I(x^5)      .
I(x^6)      .
I(x^7)      .
I(x^8)      .
I(x^9)      .
I(x^10)     .
```

最終模型選擇 4 個變數為最佳的模型，Y = 1.212561547 + 2.808334597*X + 2.696012706*X^2 + 3.940388336*X^3 + 0.006666267*X^4

Lasso 用最佳化的手法，對係數權重之和下限制，使得各變數隨著 lambda 上升時，其係數會有所收縮，所得到的最佳模式和(c)、(d)小題不同。

(f) 假設 beta0=4、bata7 = 7，epsilon 和 X 則沿用前面所產生的，得到的 Y 如下：

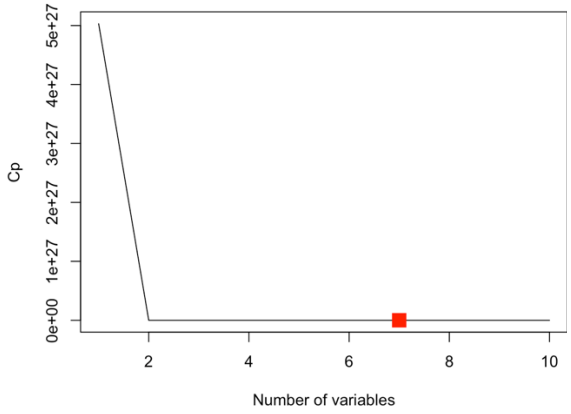
```
> Y <- b0 + b7 * X^7 + epsilon
> Y
  [1]  3.238196e+00  4.419325e+00 -6.087998e+01  4.517514e+00  3.368133e+00  4.572745e+00 -2.162706e+01
  [8]  2.336059e+00 -2.059314e+02  5.127443e+00  1.651609e+01  6.114857e+00  3.535561e+00  1.009172e+01
 [15]  5.786131e+00  4.077145e+00 -2.931709e-02  4.741811e+00  9.327499e+00  4.993832e+00  3.330960e+00
 [22] -1.661729e+03  3.521967e+00 -1.340278e+03 -3.170148e+01  3.078076e+00  3.740304e+00  2.503807e-01
 [29]  3.636017e+00  2.792531e+00 -5.211219e+02  4.633436e+00  7.600807e+00  3.318220e+00 -6.316029e+01
 [36] -8.491586e+01  4.501044e+00 -3.601196e+02  4.664644e+00  4.605813e+00  1.484022e+01  1.745379e+00
 [43]  1.585469e-01  7.443118e+00 -1.188265e+00  5.065879e+00  4.530907e+00  4.100409e+00  4.878609e+00
 [50]  4.450182e+00  3.597298e+00  3.746849e+00  6.790289e+01  1.431440e+03  3.192623e+00  2.980290e+00
 [57]  1.202725e+00  5.235434e+00  2.472135e+00  2.421236e+00 -2.647450e+01  3.421816e+00  2.616356e+00
 [64]  3.304282e+00  1.453779e+01  3.086165e+01  5.775657e+00  2.490002e+00  4.308870e+00 -3.013187e+01
 [71]  2.870746e+00 -8.218920e+01 -5.604145e+00  5.061858e+00 -1.106672e+02  2.662841e+01 -1.045581e+02
 [78]  3.672039e+00 -4.997980e+00  1.373407e+02  4.605649e+00  4.643310e+00  3.294743e+00  4.480759e+00
 [85]  4.641738e+00  6.068227e+00  4.075622e+00  1.867571e+03 -1.813411e+01  4.046361e+00  4.097847e+00
 [92]  4.995266e+00  3.318849e+00  2.195778e+01  2.716016e+00 -7.766329e+01 -1.911669e+02  5.817637e+00
 [99]  2.897917e+00  3.221481e+01
```

接著用 exhaustive 選出最佳的 model，結果如下：

```
Subset selection object
Call: regsubsets.formula(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) +
    I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10), data = data,
    nvmax = 10)
10 Variables  (and intercept)
       Forced in Forced out
x          FALSE      FALSE
I(x^2)     FALSE      FALSE
I(x^3)     FALSE      FALSE
I(x^4)     FALSE      FALSE
I(x^5)     FALSE      FALSE
I(x^6)     FALSE      FALSE
I(x^7)     FALSE      FALSE
I(x^8)     FALSE      FALSE
I(x^9)     FALSE      FALSE
I(x^10)    FALSE      FALSE
1 subsets of each size up to 10
Selection Algorithm: exhaustive
         x   I(x^2) I(x^3) I(x^4) I(x^5) I(x^6) I(x^7) I(x^8) I(x^9) I(x^10)
1  ( 1 )  " " " "    " "    " "    " "    " "    "*"    " "    " "    " "
2  ( 1 )  "*" " "    " "    " "    " "    " "    "*"    " "    " "    " "
3  ( 1 )  "*" " "    " "    " "    " "    " "    "*"    " "    " "    "*"
4  ( 1 )  "*" " "    " "    " "    " "    " "    "*"    "*"    " "    "*"
5  ( 1 )  "*" " "    " "    " "    " "    " "    "*"    "*"    "*"    "*"
6  ( 1 )  "*" " "    "*"    " "    " "    " "    "*"    "*"    "*"    "*"
7  ( 1 )  "*" " "    " "    "*"    "*"    " "    "*"    "*"    "*"    "*"
8  ( 1 )  "*" "*"    " "    " "    "*"    "*"    "*"    "*"    "*"    "*"
9  ( 1 )  "*" "*"    "*"    " "    "*"    "*"    "*"    "*"    "*"    "*"
10  ( 1 ) "*" "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"
```

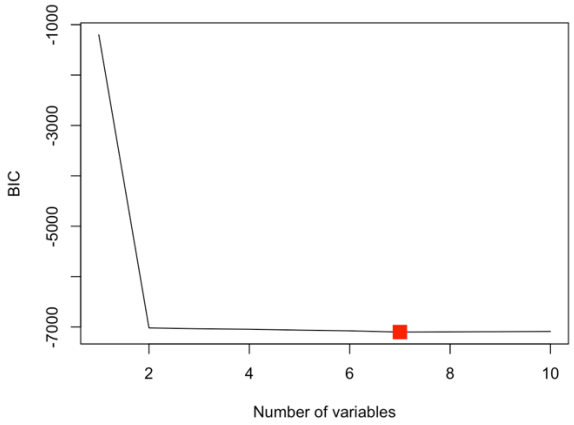
分別用 Cp、BIC、Adj R^2 來衡量模型的好壞

- Cp：最佳模式為 **7 個變數**時，下面左圖為選擇不同數量的變數分別對應的 Cp，由圖可知 Cp 最小的為選擇 7 個變數的模式(如標示的正方形所示) 。左圖為最佳模式時各變數的係數。



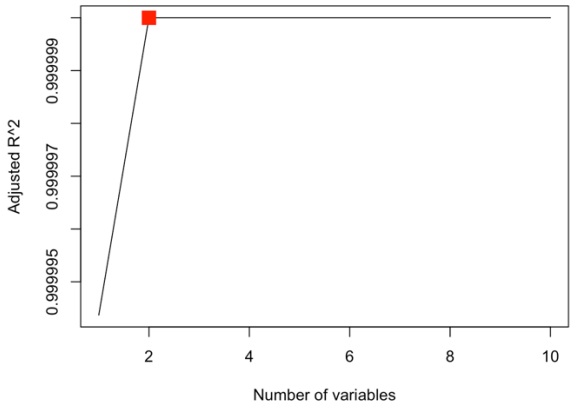

```
> coef(regfit, which.min(regsum$cp))
  (Intercept)             x        I(x^4)        I(x^5)        I(x^7)        I(x^8)        I(x^9)       I(x^10
 4.000000e+00  1.000000e+00 -1.043887e-13 -2.854522e-13  7.000000e+00  3.807603e-14 -1.885991e-14 -7.179153e-1
```

- BIC：最佳模式為 **7 個變數**時，下面左圖為選擇不同數量的變數分別對應的 BIC，由圖可知 BIC 最小的為選擇 7 個變數的模式(如標示的正方形所示) 。左圖為最佳模式時各變數的係數。




```
> coef(regfit, which.min(regsum$bic))
  (Intercept)             x        I(x^4)        I(x^5)        I(x^7)        I(x^8)        I(x^9)       I(x^10)
 4.000000e+00  1.000000e+00 -1.043887e-13 -2.854522e-13  7.000000e+00  3.807603e-14 -1.885991e-14 -7.179153e-15
```

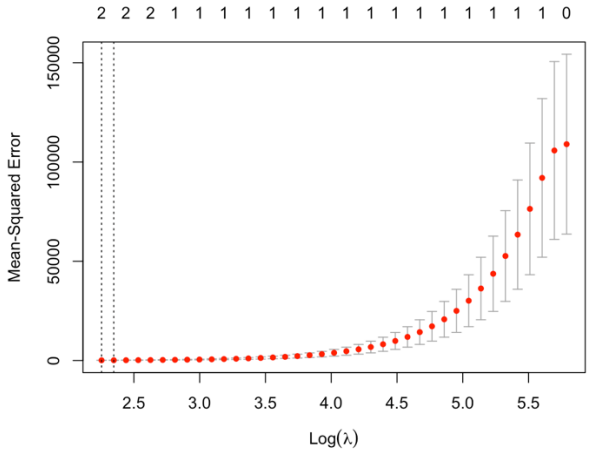
- Adj R^2：最佳模式為 **2 個變數**時，下面左圖為選擇不同數量的變數分別對應的 Adj R^2，由圖可知 Adj R^2 最小的為選擇 2 個變數的模式(如標示的正方形所示) 。左圖為最佳模式時各變數的係數。




```
> coef(regfit, which.max(regsum$adjr2))
(Intercept)           x      I(x^7)
          4           1           7
```

接著用 Lasso 選最佳模式，

最佳模式為選擇 2 個變數時，Y = 3.5845740 + 0.3049209*X^5 + 6.7405215*X^7




```
11 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) 3.5845740
x           .
I(x^2)      .
I(x^3)      .
I(x^4)      .
I(x^5)      0.3049209
I(x^6)      .
I(x^7)      6.7405215
I(x^8)      .
I(x^9)      .
I(x^10)     .
```

歸納以上結果可得知，用 exhaustive 和 lasso 選出的最佳模式不同。

10.

(a) 使用 set.seed()以確保重覆執行時，每次 random 產生的數值相同

將 b4, b8, b12, b16 設為 0

```
> b
 [1]  1.49409344  0.45880772  0.37658125  0.00000000  0.53626544 -0.86799317  0.41596265  0.00000000 -0.29416922 -0.08049533  0.29091489
[12]  0.00000000 -1.31362925 -0.59419259 -1.03572030  0.00000000 -0.66306385  2.19314507  0.99994043  0.62895821
```

(b) 使用 set.seed()以確保重覆執行時，每次 random 產生的數值相同

生成 1 到 1000 的數列，並隨機選出 100 個數為 training set 的 index

```
> train
  [1] 491 649 330 368 460 439 584 438 423 511 723 344 271 143 938 455 998 930 536 525 351 392 622 775 231 667 342 338 797 285 889 604  50 877 666
 [36] 101 890  33 647 966 946 670 324  93 114 600 307 694 754 288 267 335 417 986 969 347  42 334  13 948 361 729 712 656 373  26 857 209  48 663
 [71] 527 878 982 482 317 132 739 270  35 266  74 570 679 912 988 543 906 965 435 437 613 621  39 630  89 530 899 554 394 254
```

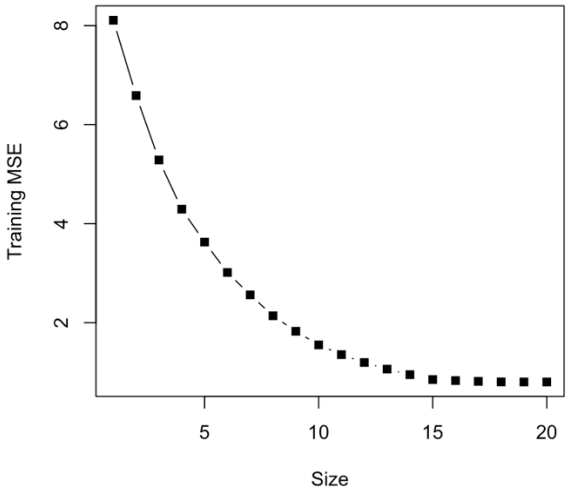
剩下的 900 個數為 test set

(c) 用 training set 配合 regsubsets 找出最佳模式，由下圖可觀察到選取參數量(由 1 到 20)對應到應該考慮哪些變數(“*”)

```
Selection Algorithm: exhaustive
          x.1 x.2 x.3 x.4 x.5 x.6 x.7 x.8 x.9 x.10 x.11 x.12 x.13 x.14 x.15 x.16 x.17 x.18 x.19 x.20
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 )  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  " "  " "  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"
20 ( 1 )  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"  "*"
```

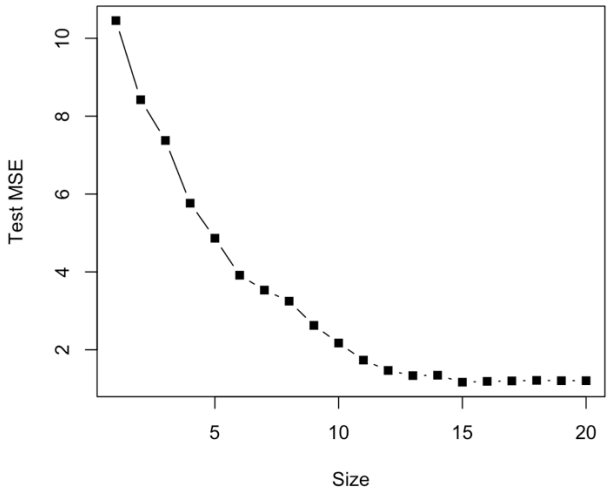
算出 training set 的 MSE，並畫在圖上

```
> val.errors
 [1] 8.1069839 6.5856900 5.2855355 4.2916796 3.6269502 3.0140846 2.5619472 2.1400355 1.8267934 1.5510128
[11] 1.3552310 1.1960308 1.0627694 0.9502795 0.8510202 0.8315368 0.8144669 0.8041766 0.8031131 0.8029937
```



(d) 用(c)的 model 算出 test set 的 MSE

```
> val.errors
 [1] 10.454865  8.421752  7.376872  5.765952  4.864542  3.914507  3.533019  3.248033  2.627321  2.172464
[11]  1.735551  1.467692  1.336041  1.346428  1.166397  1.188057  1.199077  1.213208  1.205803  1.206531
```



比較(c)和(d)，可觀察到同個 model，使用 training data 和 testing data 在不同 size 之下 MSE 的變動不同。

(e) 使用 test set，選擇 **15 個變數**的 model 的 MSE 最小，也可以在 (d) 的圖觀察到 size=15 時 MSE 最小。比較 training data 和 testing data 的 MSE，可發現因為 model 是用 training data 建的，所以模型的 MSE 曲線在 size 的變動下比 testing 的曲線來的 smooth。

```
> which.min(val.errors)
[1] 15
```

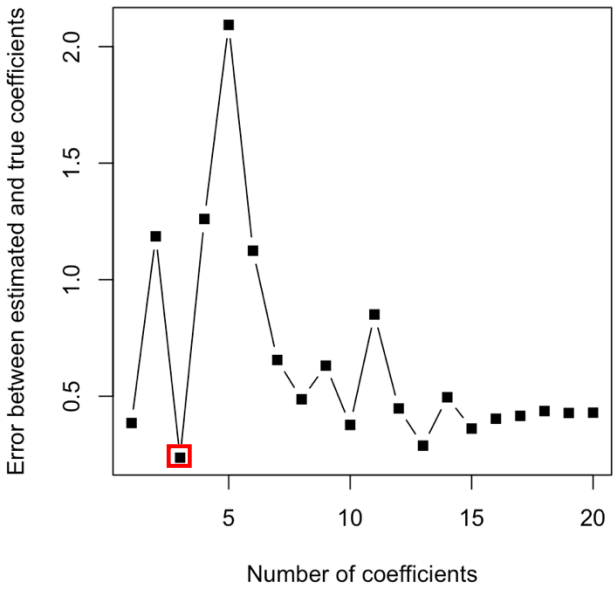
(f)

以下為取 15 個變數下(testing data 的 MSE 最小)、各變數的權重，可觀察到 X1 的權重是最大的。

```
> coef(regfit, which.min(val.errors))
(Intercept)         x.1         x.2         x.3         x.5         x.6         x.7         x.9
 -0.1015765   1.4343931   0.3622810   0.3739962   0.7015567  -0.9156748   0.5017797  -0.3699588
       x.11        x.13        x.14        x.15        x.17        x.18        x.19        x.20
  0.3546565  -1.3413124  -0.5999548  -1.0863642  -0.4457403   2.0864235   1.0013901   0.6758114
```

(g)

三個變數時，估計係數和實際係數的差值最小，然而，由(e)小題可知，選擇 15 個變數的 model MSE 最小，因此，真實係數最擬合的模型不一定 test set 的 MSE 最小。




```
> val.errors
 [1] 0.3848902 1.1859396 0.2359359 1.2604658 2.0932859 1.1244009 0.6554652 0.4867460 0.6312560 0.3767334
[11] 0.8503808 0.4474090 0.2878472 0.4952016 0.3609914 0.4033896 0.4156780 0.4360897 0.4279830 0.4292381
```

R 程式碼

```
# 8.
# (a)
set.seed(10)
X <- rnorm(100)
epsilon <- rnorm(100)
# (b)
b0 <- 4
b1 <- 3
b2 <- 2
b3 <- 1
Y <- b0+b1*X+b2*X^2+b3*X^3+epsilon
Y
# (c)
install.packages("leaps")
library(leaps)
data <- data.frame(y = Y, x = X)
regfit <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10),
data = data, nvmax = 10)
regsum <- summary(regfit)
regsum
# ?formula
names(regsum)
# cp
plot(regsum$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(which.min(regsum$cp), reg.summary$cp[which.min(regsum$cp)], col = "red", cex = 2, pch = 15)
coef(regfit, which.min(regsum$cp))
# bic
plot(regsum$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(regsum$bic), regsum$bic[which.min(regsum$bic)], col = "red", cex = 2, pch = 15)
coef(regfit, which.min(regsum$bic))
# adjr2
plot(regsum$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(which.max(regsum$adjr2), regsum$adjr2[which.max(regsum$adjr2)], col = "red", cex = 2, pch = 15)
coef(regfit, which.max(regsum$adjr2))

# (d)
# forward
regfit_fwd <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) +
I(x^10), data = data, nvmax = 10, method = "forward")
regsum.fwd <- summary(regfit_fwd)
regsum.fwd
```

```
# Cp
plot(regsum.fwd$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(which.min(regsum.fwd$cp), regsum.fwd$cp[which.min(regsum.fwd$cp)], col = "red", cex = 2, pch =
17)
coef(regfit_fwd, which.min(regsum.fwd$cp))
# BIC
plot(regsum.fwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(regsum.fwd$bic), regsum.fwd$bic[which.min(regsum.fwd$bic)], col = "red", cex = 2, pch
= 17)
coef(regfit_fwd, which.min(regsum.fwd$bic))
# adjr2
plot(regsum.fwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(which.max(regsum.fwd$adjr2), regsum.fwd$adjr2[which.max(regsum.fwd$adjr2)], col = "red", cex =
2, pch = 17)
coef(regfit_fwd, which.max(regsum.fwd$adjr2))

# backward
regfit_bwd <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) +
I(x^10), data = data, nvmax = 10, method = "backward")
regsum.bwd <- summary(regfit_bwd)
regsum.bwd
# Cp
plot(regsum.bwd$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(which.min(regsum.bwd$cp), regsum.bwd$cp[which.min(regsum.bwd$cp)], col = "red", cex = 2, pch
= 15)
coef(regfit_bwd, which.min(regsum.bwd$cp))
# BIC
plot(regsum.bwd$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(regsum.bwd$bic), regsum.bwd$bic[which.min(regsum.bwd$bic)], col = "red", cex = 2,
pch = 15)
coef(regfit_bwd, which.min(regsum.bwd$bic))
# adjr2
plot(regsum.bwd$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(which.max(regsum.bwd$adjr2), regsum.bwd$adjr2[which.max(regsum.bwd$adjr2)], col = "red", cex
= 2, pch = 15)
coef(regfit_bwd, which.max(regsum.bwd$adjr2))

# (e)
library(glmnet)
xmat <- model.matrix(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) +
I(x^10), data = data)[, -1]
xmat
```

```
# lasso
cv <- cv.glmnet(xmat, Y, alpha = 1)
plot(cv)
names(cv)
cv$lambda.min
cv$lambda.1se
coef(cv, s = "lambda.min")

# (f)

b7 <- 7
Y <- b0 + b7 * X^7 + epsilon
Y
data <- data.frame(y = Y, x = X)
regfit <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) + I(x^10),
data = data, nvmax = 10)
regsum <- summary(regfit)
regsum
# cp
plot(regsum$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(which.min(regsum$cp), reg.summary$cp[which.min(regsum$cp)], col = "red", cex = 2, pch = 15)
coef(regfit, which.min(regsum$cp))
# bic
plot(regsum$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(which.min(regsum$bic), regsum$bic[which.min(regsum$bic)], col = "red", cex = 2, pch = 15)
coef(regfit, which.min(regsum$bic))
# adjr2
plot(regsum$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l")
points(which.max(regsum$adjr2), regsum$adjr2[which.max(regsum$adjr2)], col = "red", cex = 2, pch = 15)
coef(regfit, which.max(regsum$adjr2))

# lasso
xmat <- model.matrix(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9) +
I(x^10), data = data)[, -1]
xmat
cv <- cv.glmnet(xmat, Y, alpha = 1)
plot(cv)
names(cv)
cv$lambda.min
cv$lambda.1se
coef(cv, s = "lambda.min")
```

```
# 10.
# (a)
set.seed(10)
x <- matrix(rnorm(1000 * 20), 1000, 20)
x
b <- rnorm(20)
b
b[4] <- 0
b[8] <- 0
b[12] <- 0
b[16] <- 0
b
epsilon <- rnorm(1000)
y <- x %*% b + epsilon
y
# (b)
set.seed(10)
train <- sample(seq(1000), 100, replace = FALSE)
test <- -train
x.train <- x[train, ]
x.test <- x[test, ]
y.train <- y[train]
y.test <- y[test]
# (c)
data.train <- data.frame(y = y.train, x = x.train)
regfit <- regsubsets(y ~ ., data = data.train, nvmax = 20)
regsum <- summary(regfit)

train.mat <- model.matrix(y ~ ., data = data.train, nvmax = 20)
val.errors_train <- rep(NA, 20)
for (i in 1:20) {
   # ?coef()
   coef <- coef(regfit, id = i)
   pred <- train.mat[, names(coef)] %*% coef
   val.errors_train[i] <- mean((pred - y.train)^2)
}
plot(val.errors_train, xlab = "Size", ylab = "Training MSE", pch = 15, type = "b")
val.errors_train

# (d)
data.test <- data.frame(y = y.test, x = x.test)
test.mat <- model.matrix(y ~ ., data = data.test, nvmax = 20)
```

```
val.errors_test <- rep(NA, 20)
for (i in 1:20) {
   coef <- coef(regfit, id = i)
   pred <- test.mat[, names(coef)] %*% coef
   val.errors_test[i] <- mean((pred - y.test)^2)
}
plot(val.errors_test, xlab = "Size", ylab = "Test MSE", pch = 15, type = "b")
val.errors_test

# (e)
which.min(val.errors_test)
# (f)
coef(regfit, which.min(val.errors_test))

# (g)
val.errors <- rep(NA, 20)
x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
for (i in 1:20) {
   coefi <- coef(regfit, id = i)
   val.errors[i] <- sqrt(sum((b[x_cols %in% names(coefi)] - coefi[names(coefi) %in% x_cols])^2) +
sum(b[!(x_cols %in% names(coefi))])^2)
}
plot(val.errors, xlab = "Number of coefficients", ylab = "Error between estimated and true coefficients", pch
= 15, type = "b")
val.errors
min(val.errors)
```