МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

ЗВІТ

до індивідуального завдання №5

з дисципліни «Моделі статистичного навчання»

Виконав
студент групи ПМіМ-12:
Бордун Михайло

Перевірив:
Проф. Заболоцький Т. М.

Львів – 2021

Хід виконання

1. На основі згенерованих даних застосуємо вибір найкращої підмножини.

1.1 Використовуючи функцію rnorm() згенеруйте предиктор X довжиною n = 100, та вектор залишків ε такої ж довжини n = 100.

```
set.seed(1)
x = rnorm(100)
eps = rnorm(100)
```

1.2 Згенеруйте вектор залежних змінних Y довжини n = 100 відповідно до моделі

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon,$$

де β_0, β_1, β_2 і β_3 константи на ваш вибір.

```
betas = runif(5, min=-5, max=5)
print("Beta-values list:")
print(betas)

y = betas[1] + betas[2] * x +
 betas[3] * x^2 + betas[4] * x^3 + eps
```

```
[1] "Beta-values list:"
[1]  1.588776 -3.149300  4.543781  3.978485  4.436971
```

Визначення значень β_i відбувається випадково (в минулому пункті визначено seed як 1), в межах від -5 до 5 будь-які дробові числа. Останнє значення – це значення $\beta_{7,}$ яке нам буде необхідне в наступних пунктах.

1.3 Використовуючи функцію regsubsets() виберіть найкращу модель методом вибору найкращої підмножини з множини предикторів X, X^2, . . ., X^{10}. Яка модель найкраща за показниками C_p, BIC і скорегований R^2? Наведіть декілька графіків на підтвердження своєї відповіді та вкажіть оцінки коефіцієнтів найкращої моделі.

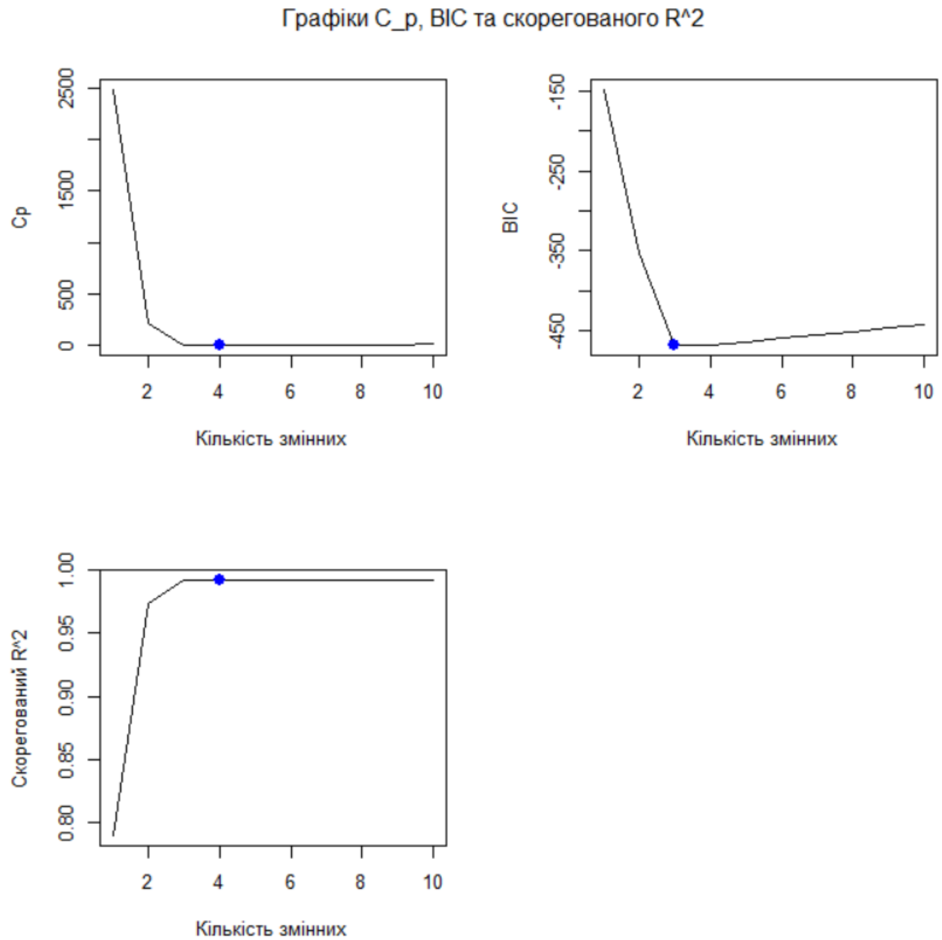
```
data.frame = data.frame(y = y, x = x)

BestModelSelection = function(method_, mtext_) {
    cat("\n")
    reg.fit = 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.frame,
     nvmax = 10, method = method_)
    reg.summary = summary(reg.fit)
    print(reg.summary)

    par(mfrow = c(2, 2))

    plot(reg.summary$cp, xlab = "Кількість змінних", ylab = "Cp", type = "l")
    points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)],
    col = "blue", cex = 2, pch = 20)

    plot(reg.summary$bic, xlab = "Кількість змінних", ylab = "BIC", type = "l")
    points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)],
    col = "blue", cex = 2, pch = 20)

    plot(reg.summary$adjr2, xlab = "Кількість змінних", ylab = "Скорегований R^2", type
= "l")
    points(which.max(reg.summary$adjr2),
reg.summary$adjr2[which.max(reg.summary$adjr2)],
    col = "blue", cex = 2, pch = 20)

    mtext(mtext_, side = 3, line = -2, outer = TRUE)

    cat("\n")

    point.a = which.min(reg.summary$cp)
    point.b = which.min(reg.summary$bic)
    point.c = which.max(reg.summary$adjr2)
```

```
    if (point.a == point.b && point.a == point.c) {
      print(coef(reg.fit, point.a))
    } else if (point.a == point.b && point.a != point.c) {
      print(coef(reg.fit, point.b))
      print(coef(reg.fit, point.c))
    } else if (point.a != point.b && point.a == point.c ||
              point.a != point.b && point.b == point.c) {
      print(coef(reg.fit, point.a))
      print(coef(reg.fit, point.b))
    } else if (point.a != point.b && point.b == point.c) {
    } else {
      print(coef(reg.fit, point.a))
      print(coef(reg.fit, point.b))
      print(coef(reg.fit, point.c))
    }
}

BestModelSelection("exhaustive",
 "Графіки C_p, BIC та скорегованого R^2")
```

```
Subset selection object
Call: BestModelSelection("exhaustive",
 "Графіки C.p, BIC та скорегованого R^2")
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 ) "*" "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"    "*"
```

Як бачимо тут подано найкращий набір змінних для кожної розмірності моделі. Зірочки тут означають, що дана змінна включена у відповідну модель.

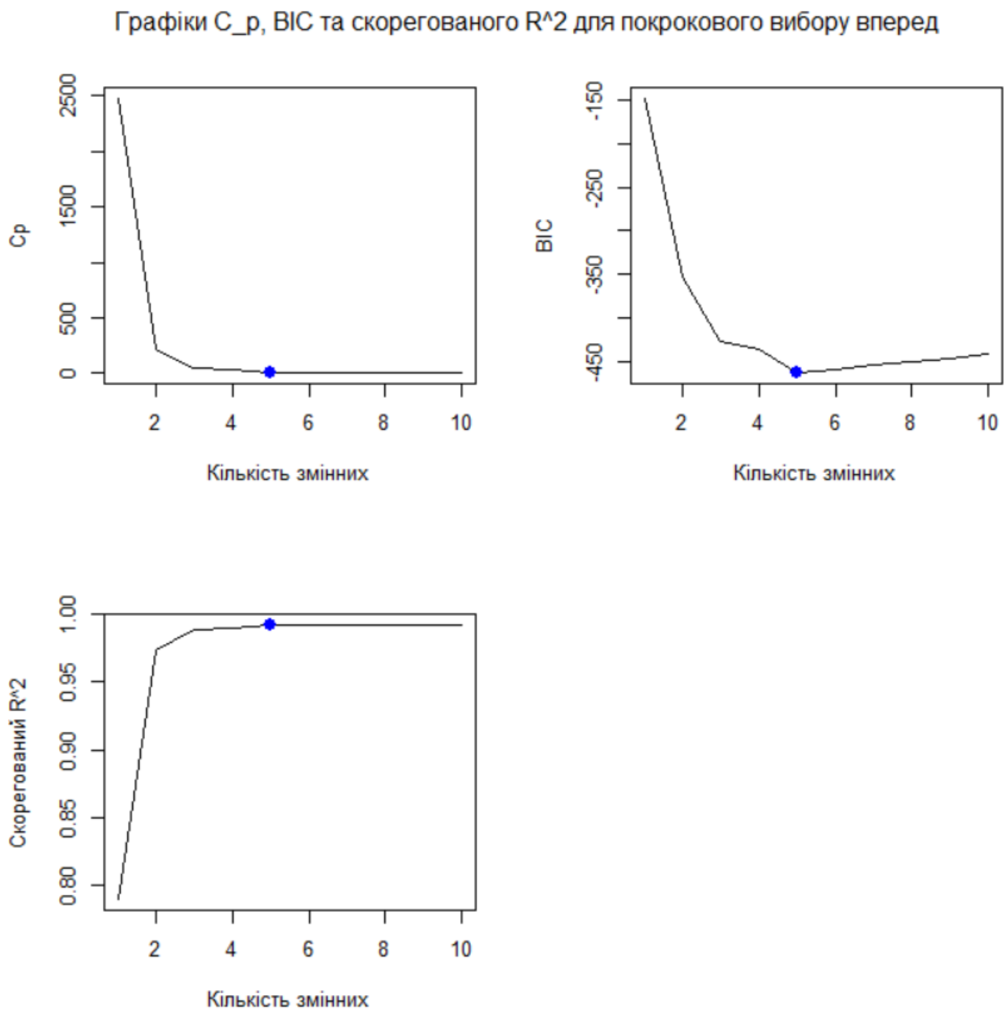
```
(Intercept)           x      I(x^2)      I(x^3)      I(x^5)
 1.66078384 -2.76184440  4.38953778  3.53645918  0.08072292
(Intercept)           x      I(x^2)      I(x^3)
   1.650283   -3.174020    4.419990    3.996123
```



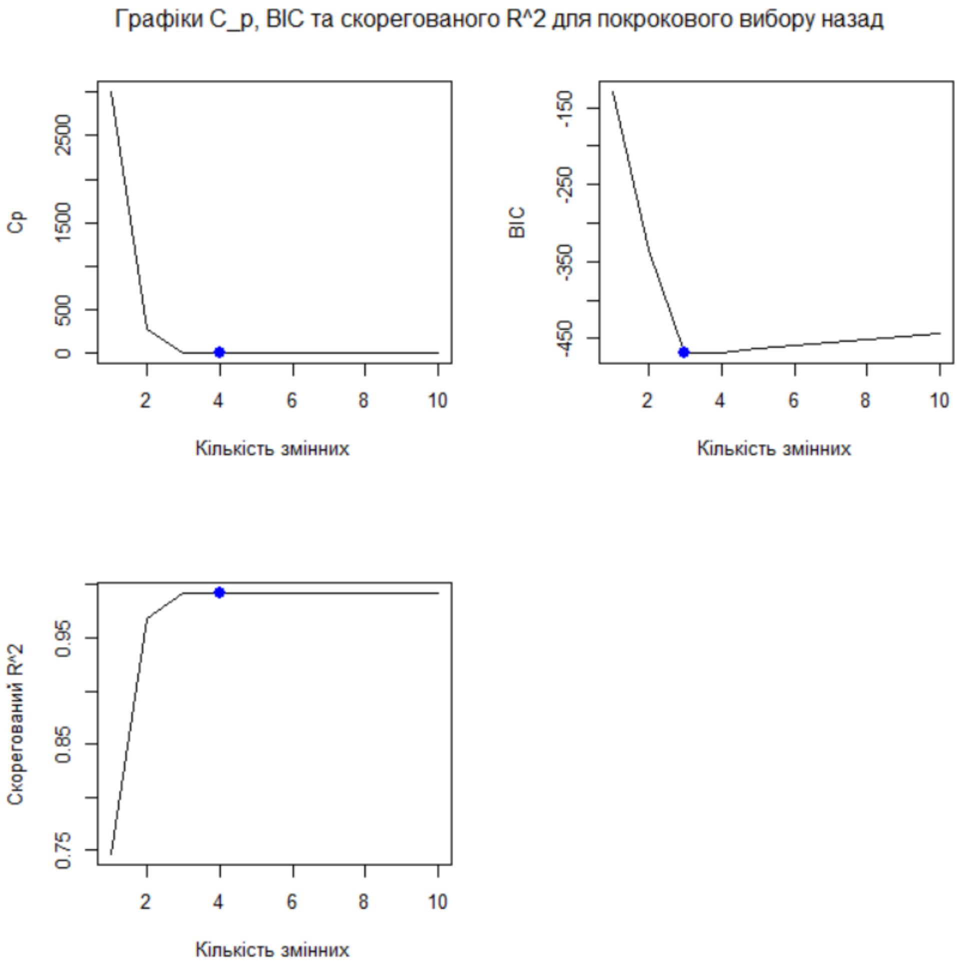
З огляду на наведені вище результати, бачимо, що найкраща модель за показниками C_p та скорегованим R^2 – це модель з 4 змінними (x, x^2, x^3, x^5). Для показника BIC бачимо, що вже найкращою буде модель зі змінними x, x^2 та x^3.

1.4 Повторіть 1.3, використовуючи методи покрокового вибору вперед та назад. Порівняйте отримані результати з 1.3.

```
  (Intercept)             x        I(x^2)        I(x^3)        I(x^5) 
 1.6602897552 -2.7530199571  4.3904564519  3.5177817906  0.0891021995 
       I(x^7) 
-0.0009812316 
```




```
 (Intercept)            x       I(x^2)       I(x^3)       I(x^9)
 1.668012452 -2.917394527  4.377275549  3.798040727  0.001290827
(Intercept)           x      I(x^2)      I(x^3)
   1.650283   -3.174020    4.419990    3.996123
```



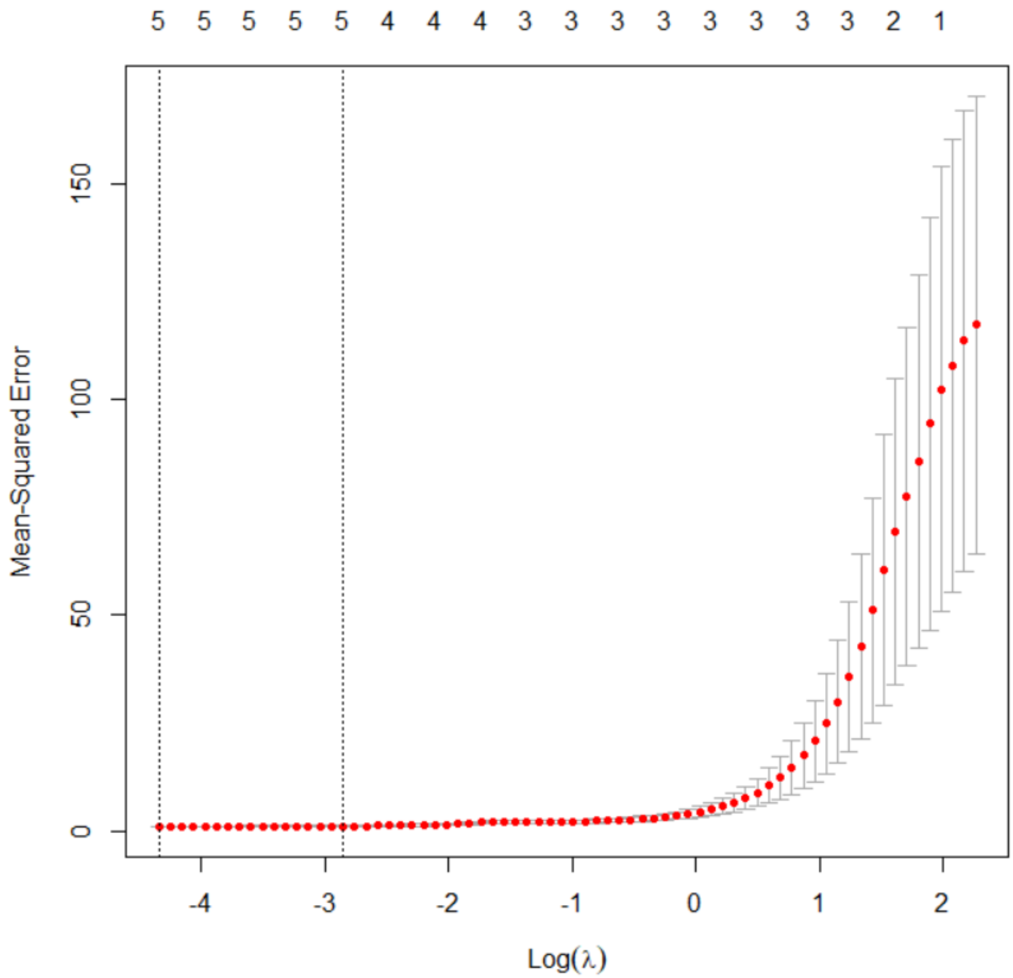
Аналізуючи наведені результати, можна сказати, що кожен метод дає інший результат, і тільки для методу покрокового вибору вперед для всіх показників найкращою буде модель з 5-ма змінними (x, x^2, x^3, x^5, x^7).

Для методу покрокового вибору назад результат дуже подібних до методу вибору найкращої підмножини, оскільки для кожного показника найкраща модель має таку ж кількість змінних, але вони відрізняються: для покрокового вибору назад найкраща модель за показниками C_p та скорегованим R^2 є зі змінними x, x^2, x^3, x^9. Для показника BIC бачимо, що вже найкращою буде модель зі змінними x, x^2 та x^3.

1.5 Пристосуйте ласо модель до згенерованих даних, використовуючи X, X^2, . . ., X^{10} як предиктори . Використайте перехресну перевірку для вибору значення λ. Побудуйте графіки помилки перехресної перевірки як функції від λ. Наведіть отримані оцінки коефіцієнтів моделі та обгрунтуйте отримані результати.

```
Lasso = function() {
    cat("\n")
    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.frame)[, -1]

    cv.out = cv.glmnet(xmat, y, alpha = 1)
    par(mfrow = c(1, 1))

    bestlam = cv.out$lambda.min
    print(paste('Min lambda: ', bestlam))
    plot(cv.out)

    out = glmnet(xmat, y, alpha = 1)
    lasso.coefs = predict(out, s = bestlam, type = "coefficients")[1:11, ]
    print(lasso.coefs[lasso.coefs != 0])
}
Lasso()
```

```
[1] "Min lambda:  0.0129972236622867"
(Intercept)           x      I(x^2)      I(x^3)      I(x^4)      I(x^5)
 1.70350206 -2.55037422  4.25913284  3.38384904  0.02866323  0.10039894
```



З рисунка видно, що при додатних значеннях логарифма лямбли помилка стрімко зростає. В результаті бачимо, що методом лассо ми знайшли мінімальну в сенсі помилки лямбду, яка дорівнює 0.013 і з допомогою неї знайшли найкращу модель, яка включає в себе 5 змінних (x, x^2, x^3, x^4, x^5).

1.6 Згенеруйте вектор залежних змінних *Y* відповідно до моделі

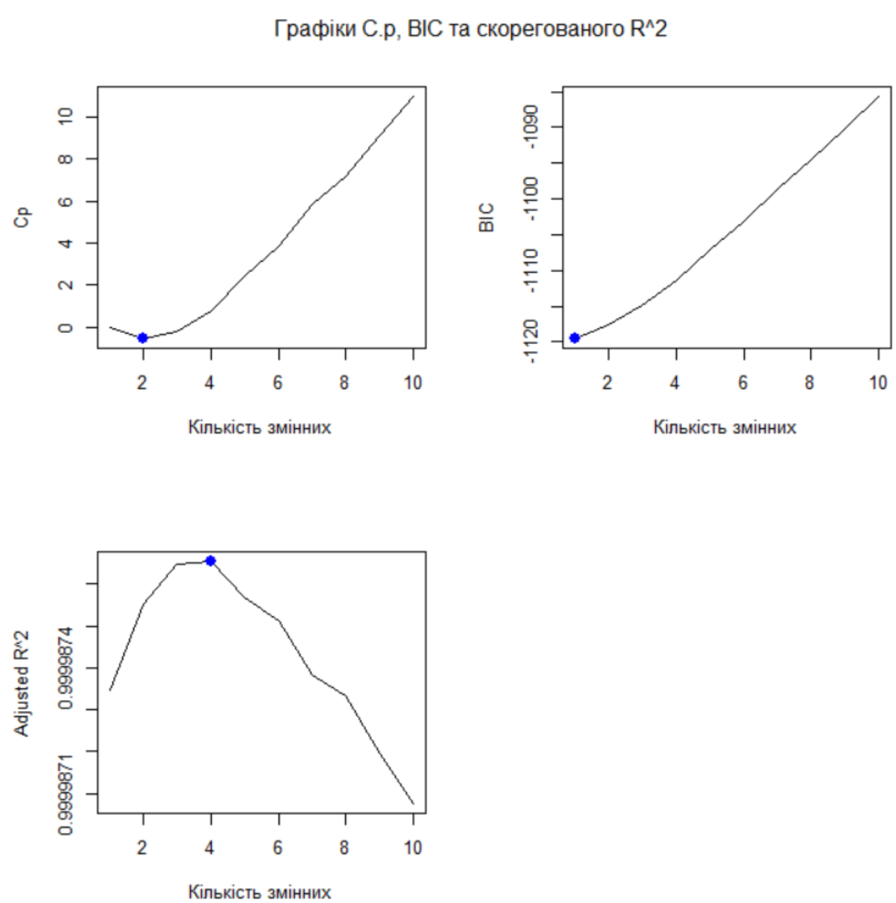
$$Y = \beta_0 + \beta_7 X^7 + \varepsilon,$$

і застосуйте метод найкращого вибору підмножини і ласо. Обгрунтуйте отримані результати.

```
y = betas[1] + betas[5] * x^7 + eps
```

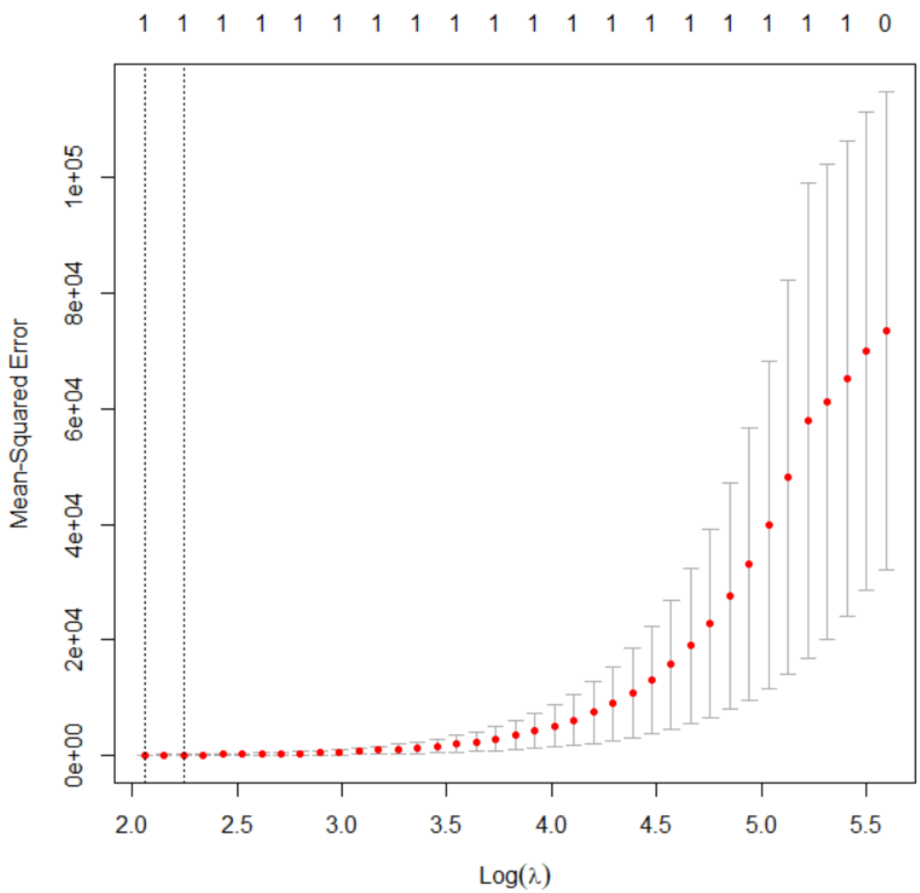
```r
data.frame = data.frame(y = y, x = x)

BestModelSelection("exhaustive",
 "Графіки C.p, BIC та скорегованого R^2")
Lasso()
```

```
(Intercept)      I(x^2)      I(x^7)
  1.6592665  -0.1417084   4.4385257
(Intercept)      I(x^7)
   1.547716    4.437741
(Intercept)           x      I(x^2)      I(x^3)      I(x^7)
  1.6650285   0.2914016  -0.1617671  -0.2526527   4.4461043

[1] "Min lambda:  7.84052238330809"
(Intercept)      I(x^7)
   2.093672    4.308379
```



Враховуючи наведені вище результати, бачимо, що найкраща модель для всіх показників має різну кількість змінних, так: за показниками C_p – це модель з 2 змінними (x^2, x^7). скорегованим R^2 модель з 4 змінними (x, x^2, x^3, x^7). Для показника BIC бачимо, що вже найкращою буде модель зі змінною x^7.

Оскільки β_7 = 4.437, то можемо з впевненістю сказати, що метод найкращого вибору підмножини з використанням BIC визначає найбільш точну модель з однією змінною.



Щодо методу лассо, то ми знайшли мінімальну в сенсі помилки лямбду, яка дорівнює 7.84 і з допомогою неї знайшли найкращу модель, яка включає одну змінну x^7. Проте точність моделі є гіршою у порівнянні з методом найкращого вибору підмножини з використанням показника BIC.

2. На основі даних College передбачимо кількість отриманих заяв.

```
 Private        Apps           Accept          Enroll        Top10perc
 No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
 Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
           Median : 1558   Median : 1110   Median : 434   Median :23.00
           Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
           3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
           Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
   Top25perc      F.Undergrad     P.Undergrad         Outstate
 Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
 1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
 Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
 Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
 3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
 Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
   Room.Board       Books           Personal         PhD
 Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
 1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
 Median :4200   Median : 500.0   Median :1200   Median : 75.00
 Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
 3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
 Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
    Terminal       S.F.Ratio      perc.alumni        Expend
 Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
 1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
 Median : 82.0   Median :13.60   Median :21.00   Median : 8377
 Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
 3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
 Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
   Grad.Rate
 Min.   : 10.00
 1st Qu.: 53.00
 Median : 65.00
 Mean   : 65.46
 3rd Qu.: 78.00
 Max.   :118.00
```

Загальна характеристика даних College

2.1 Розбийте набір даних на навчальний та тестовий набори.

```
set.seed(1)
train = sample(1:length(Apps), 0.5 * length(Apps))
College.train = College[train, ]
College.test = College[-train, ]
```

2.2 Оцініть лінійну модель, використовуючи метод найменших квадратів на навчальному наборі, та обчисліть тестову помилку.

```
fit.lm = lm(Apps ~ ., data = College.train)
pred.lm = predict(fit.lm, College.test)
```

```
cat("\n")
print(paste("Test error: ", round(mean((pred.lm - College.test$Apps)^2), 2)))
```

```
[1] "Test error:  1135758.32"
```

2.3 Пристосуйте модель гребеневої регресії до тренувального набору, вибравши λ шляхом перехресної перевірки. Обчисліть тестову помилку.

```
train.mat = model.matrix(Apps ~ ., data = College.train)
test.mat = model.matrix(Apps ~ ., data = College.test)

grid = 10 ^ seq(10, -2, length = 100)
fit.ridge = glmnet(train.mat, College.train$Apps, alpha = 0,
 lambda = grid)
cv.ridge = cv.glmnet(train.mat, College.train$Apps, alpha = 0,
 lambda = grid)
cat("\n")
print(dim(coef(fit.ridge)))

bestlam = cv.ridge$lambda.min
print(paste('Min lambda: ', bestlam))
pred.ridge = predict(fit.ridge, s = bestlam, newx = test.mat)
print(paste("Test error: ", round(mean((pred.ridge - College.test$Apps)^2), 2)))
```

```
 19 100
"Min lambda:  0.01"
"Test error:  1134676.8"
```

З кожним вектором лямбда пов’язаний вектор коефіцієнтів гребеневої регресії, які зберігаються в матриці, до якої можна отримати доступ через coef(). У цьому випадку це 19х100 матриця (лямбди з огляду на grid() є в межах від 10^{-2} до 10^{10}).

Мінімальна в сенсі помилки лямбда дорівнює 0.01 і з допомогою неї знайшли найкращу тестову помилку, що дорівнює 1134676.8, що є кращим результатом в порівнянні з обчисленою методом найменших квадратів.

2.4 Пристосуйте модель ласо до тренувального набору, вибравши λ шляхом перехресної перевірки. Обчисліть тестову помилку. Яка кількість ненульових оцінок коефіцієнтів.

```
fit.lasso = glmnet(train.mat, College.train$Apps, alpha = 1,
 lambda = grid)
cv.lasso  = cv.glmnet(train.mat, College.train$Apps, alpha = 1,
 lambda = grid)
cat("\n")

bestlam = cv.lasso$lambda.min
print(paste('Min lambda: ', bestlam))
pred.lasso = predict(fit.lasso, s = bestlam, newx = test.mat)
print(paste("Test error: ", round(mean((pred.lasso - College.test$Apps)^2), 2)))

cat("\n")
lasso.coefs = predict(fit.lasso, s = bestlam, type = "coefficients")
print(lasso.coefs)
```

```
"Min lambda:  0.01"
"Test error:  1133422.13"
```

Мінімальна в сенсі помилки лямбда дорівнює 0.01 і з допомогою неї знайшли найкращу тестову помилку, що дорівнює 1133422.1, що є кращим результатом в порівнянні з обчисленою методом найменших квадратів та гребеневої регресії.

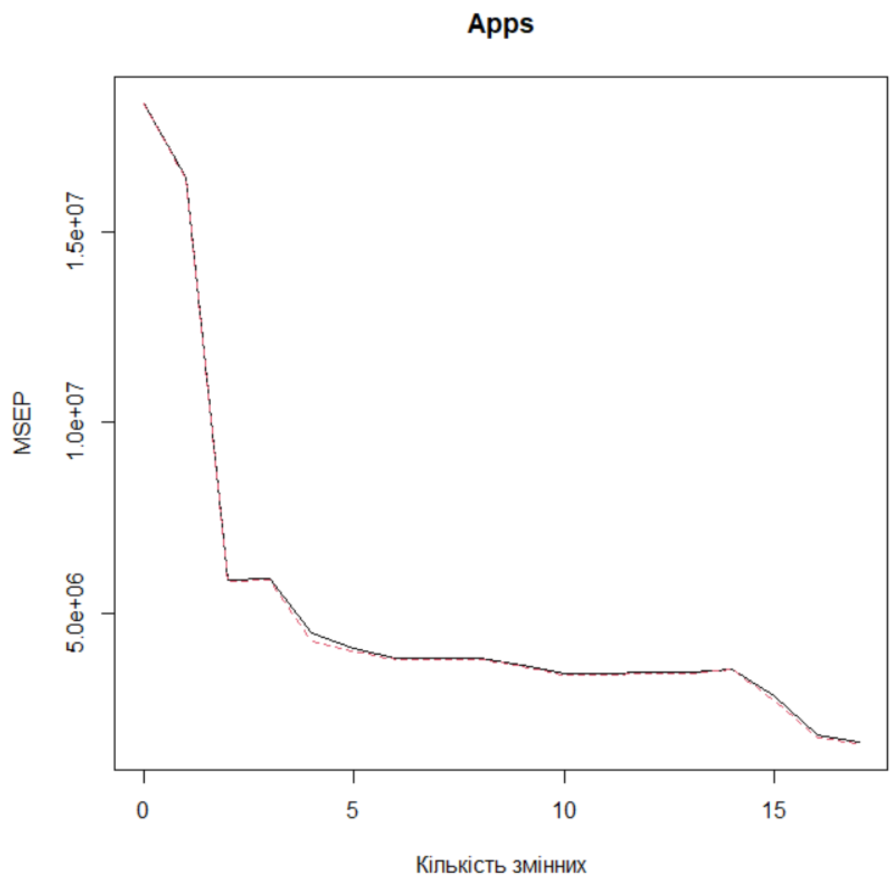
```
19 x 1 sparse Matrix of class "dgCMatrix"
                        s1
(Intercept) -7.931498e+02
(Intercept)  .
PrivateYes  -3.078903e+02
Accept       1.777242e+00
Enroll      -1.450532e+00
Top10perc    6.659456e+01
Top25perc   -2.221506e+01
F.Undergrad  8.983869e-02
P.Undergrad  1.005260e-02
Outstate    -1.082871e-01
Room.Board   2.118762e-01
Books        2.922508e-01
Personal     6.234085e-03
PhD         -1.542914e+01
Terminal     6.364841e+00
S.F.Ratio    2.284667e+01
perc.alumni  1.114025e+00
Expend       4.861825e-02
Grad.Rate    7.466015e+00
```

Також бачимо, що жоден з коефіцієнтів не дорівнює нулю.

2.5 Пристосуйте модель PCR до тренувального набору, причому M виберіть шляхом перехресної перевірки. Яке отримане значення M? Обчисліть отриману помилку тесту.

```
fit.pcr = pcr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
validationplot(fit.pcr, val.type = "MSEP", xlab = "Кількість змінних")

cat("\n")
print(paste('Min M: ', which.min(fit.pcr$validation$adj)))
pred.pcr = predict(fit.pcr, College.test, ncomp = which.min(fit.pcr$validation$adj))
print(paste("Test error: ", round(mean((pred.pcr - College.test$Apps)^2), 2)))
```

```
"Min M:  17"
"Test error:  1135758.32"
```

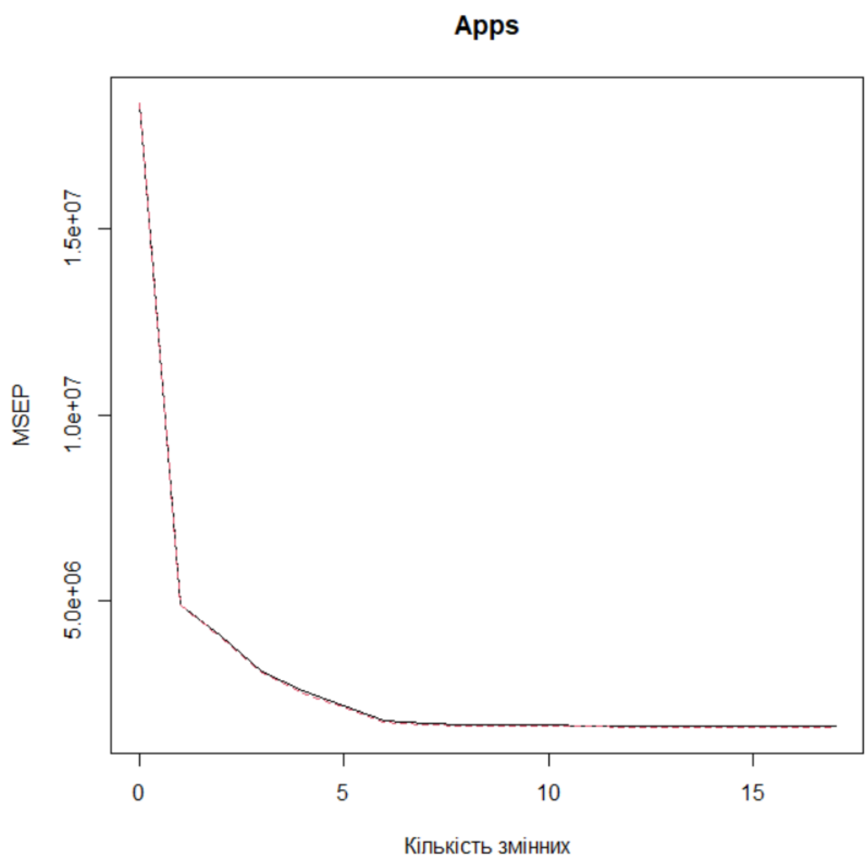


Мінімальна в сенсі помилки модель має M = 17 з тестовою помилкою, що дорівнює 1135758.3, що є ідентичним результатом в порівнянні з обчисленою методом найменших квадратів.

2.6 Пристосуйте модель PLS до тренувального набору, причому M виберіть шляхом перехресної перевірки. Яке отримане значення M? Обчисліть отриману помилку тесту.

```
fit.pls = plsr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
validationplot(fit.pls, val.type = "MSEP", xlab = "Кількість змінних")

cat("\n")
print(paste('Min M: ', which.min(fit.pls$validation$adj)))
pred.pls = predict(fit.pls, College.test, ncomp = which.min(fit.pls$validation$adj))
print(paste("Test error: ", round(mean((pred.pls - College.test$Apps)^2), 2)))
```

```
"Min M:  17"
"Test error:  1135758.32"
```



Мінімальна в сенсі помилки модель має M = 17 з тестовою помилкою, що дорівнює 1135758.3, що є ідентичним результатом в порівнянні з обчисленою методом найменших квадратів та помилкою у моделі PCR.

2.7 Прокоментуйте отримані результати. Наскільки точно ми можемо передбачити кількість отриманих заявок на коледж? Чи велика різниця між тестовими помилками, що виникають внаслідок розглянутих п’яти підходів?

```
test.mean = mean(College.test$Apps)
methods.list = c(pred.lm, pred.ridge, pred.lasso, pred.pcr, pred.pls)
```

```
GetRSqr = function(m) {
  r_result = 1 - mean((m - College.test$Apps)^2) / mean((test.mean - College.test$Apps)^2)
  return(round(r_result, 7)*100)
}

cat("\n")
print(paste("lm prediction R^2: ", GetRSqr(pred.lm), " %"))
print(paste("ridge prediction R^2: ", GetRSqr(pred.ridge), " %"))
print(paste("lasso prediction R^2: ", GetRSqr(pred.lasso), " %"))
print(paste("pcr prediction R^2: ", GetRSqr(pred.pcr), " %"))
print(paste("pls prediction R^2: ", GetRSqr(pred.pls), " %"))
```

```
"lm prediction R^2:  90.15413  %"
"ridge prediction R^2:  90.16351  %"
"lasso prediction R^2:  90.17438  %"
"pcr prediction R^2:  90.15413  %"
"pls prediction R^2:  90.15413  %"
```

Для перевірки точності передбаченя кількості отриманих заявок на коледж було виконано обчислення коефіцієнту детермінації, який і показує наскільки отримані спостереження підтверджують модель.

Як бачимо R^2 є найближчим до ідеального (90.1744 %) для моделі лассо і найнижчий для моделей PCR, PLS та лінійної (90.1541 %).

Щодо різниця між тестовими помилками, то вона є досить мала у розглянутих 5-ти підходах, а саме різниця між найбільшою помилкою та найменшою складає близько 0.2%.

3. Ми бачили, що зі збільшенням кількості предикторів, що використовуються в моделі, навчальна помилка обов'язково зменшиться, але тестова - не обов'язково. Дослідимо це на згенерованих даних.

3.1 Сформуйте набір даних з p = 20 ознаками, n = 1000 спостереженнями, і пов'язаний з ним вектор залежних змінних відповідно до моделі

$$Y = X\beta + \varepsilon,$$

де вектор β має деякі елементи, які точно дорівнюють нулю.

```
set.seed(1)
x = matrix(rnorm(1000 * 20), 1000, 20)
eps = rnorm(1000)

cat("\n")
betas = runif(20, min=-5, max=5)
print("Beta-values list:")

zero_vals_num = sample(3:10, 1)
for (i in sample(1:length(betas), zero_vals_num)) {
  betas[i] = 0
}
print(betas)

y = x %*% betas + eps
```

```
[1] "Beta-values list:"
 [1]  2.26110490  0.00000000 -4.97172271  4.41650527  0.72021482  3.83971442
 [7]  0.00000000 -2.92841800  0.00000000  0.00000000  3.76419525  2.40631339
[13]  0.99081832 -3.94591790 -3.64146732 -1.71236717  1.45211214 -0.01610153
[19] -4.62816600  0.00000000
```

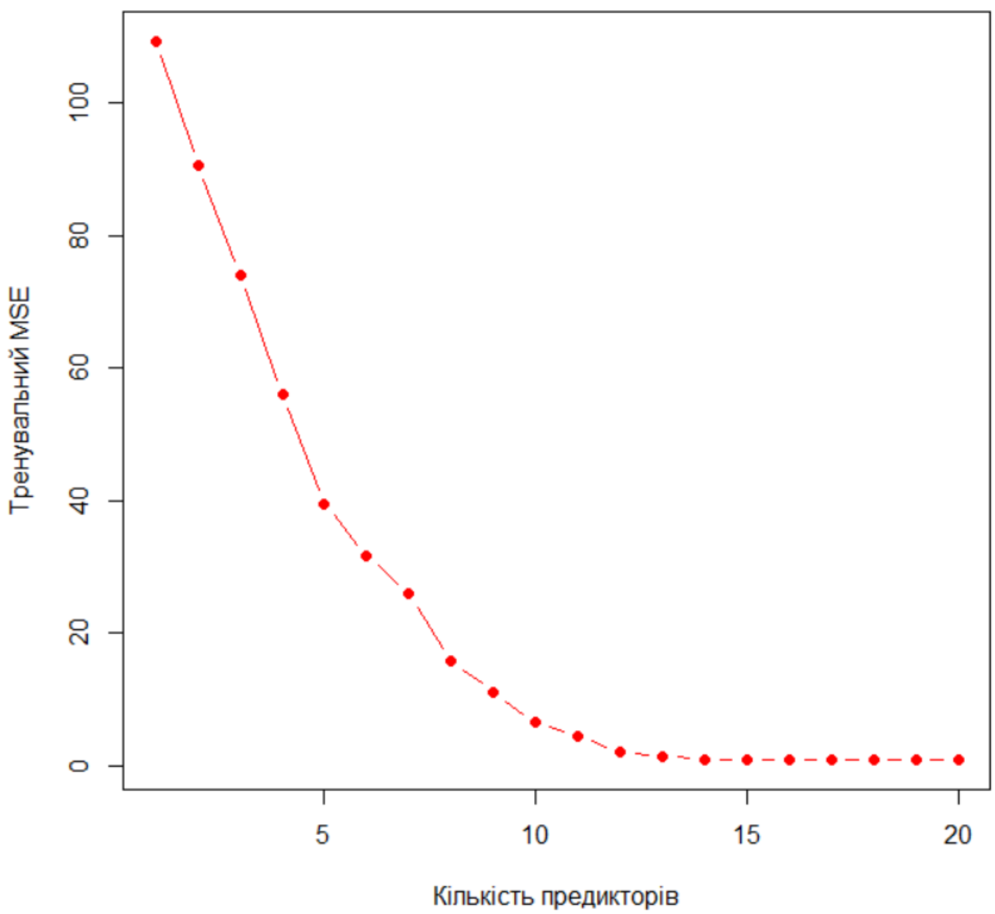
Визначення значень β_i відбувається випадково (seed дорівнює 1), в межах від -5 до 5 будь-які дробові числа. Далі також випадково береться число від 3 до 10, що позначає кількість значень вектора β, які точно дорівнюють нулю, і визначається також випадково які конкретно значення будуть занулюватись. Результат виводу вектора можна бачити вище.

3.2 Розділіть свій набір даних на навчальний набір, що містить 100 спостережень та тестовий набір, що містить 900 спостережень.

```
train = sample(1:length(eps), 100)

x.train = x[train, ]
y.train = y[train]

x.test = x[-train, ]
y.test = y[-train]
```

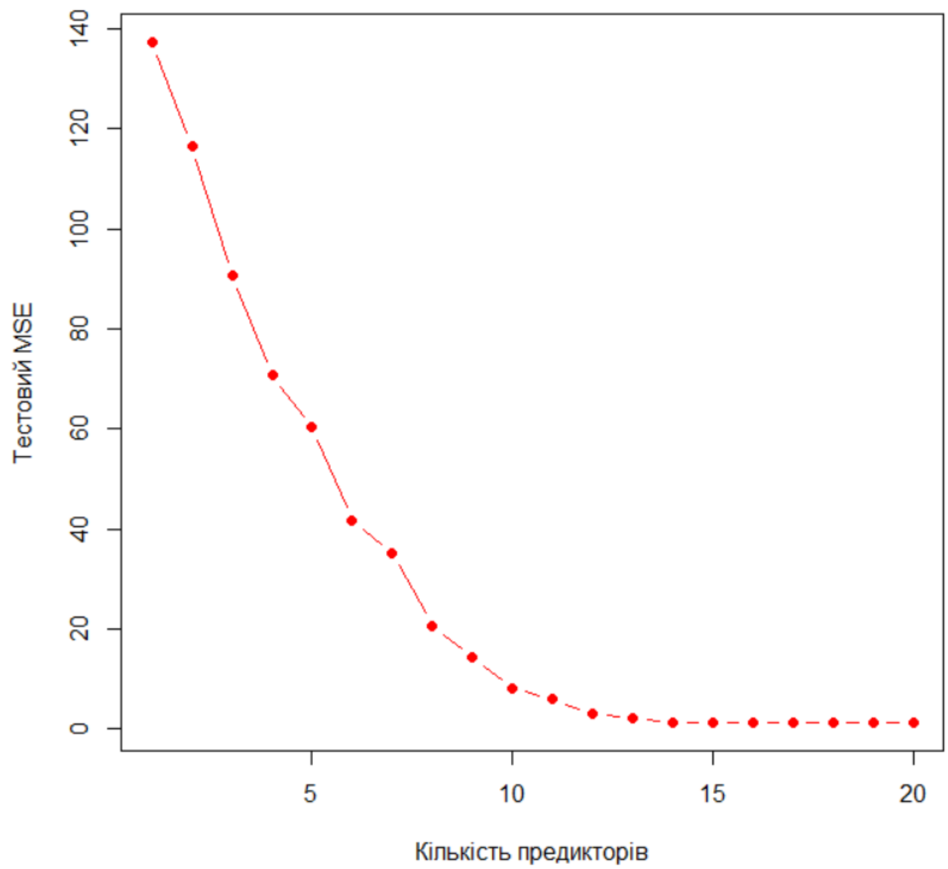
3.3 Використайте метод вибору найкращої підмножини на навчальному наборі та побудуйте графік навчального MSE, який відповідає найкращій моделі кожного розміру.

```
reg.fit = regsubsets(y ~ .,
                     data = data.frame(y = y.train, x = x.train),
                     nvmax = 20)

BestSubsetSelection = function(x_, y_, ylab_) {
    data_ = data.frame(y = y_, x = x_)
    mat = model.matrix(y ~ ., data = data_, nvmax = 20)

    val.errors = rep(0, 20)
    for (i in 1:20) {
        coef_i = coef(reg.fit, id = i)
        pred = mat[, names(coef_i)] %*% coef_i
        val.errors[i] = mean((pred - y_)^2)
    }

    plot(val.errors, xlab = "Кількість предикторів", ylab = ylab_,
    col = "red", pch = 19, type = "b")

    return(val.errors)
}

BestSubsetSelection(x.train, y.train, "Тренувальний MSE")
```



З графіка навчального MSE бачимо, що справді зі збільшенням кількості предикторів, що використовуються в моделі, навчальна помилка зменшується.

3.4 Побудуйте графік тестового MSE, який відповідає найкращій моделі кожного розміру.



3.5 Для якого розміру моделі тестовий MSE приймає мінімальне значення? Прокоментуйте отримані результати.

```
print(paste('Model size for min MSE: ', which.min(test_mse),
 ', min MSE: ', min(test_mse)))
```

```
"Model size for min MSE:  14 , min MSE:  1.16165441775166"
```

Хоч і з графіка досить складно визначити розмір моделі, де тестовий MSE приймає мінімальне значення, але це модель з 14-ма змінними, тобто модель не містить всі предиктори як, наприклад, це було з навчальною помилкою. Сам MSE в даному випадку дорівнює 1.1617.

3.6 Як співвідносяться модель, що мінімізує тестовий MSE та справжня модель, яка використовувалася для генерації даних? Прокоментуйте значення оцінок коефіцієнтів.

```
print(coef(reg.fit, which.min(test_mse)))
```

```
(Intercept)         x.1         x.3         x.4         x.5         x.6 
 0.03101094  2.35530383 -4.91704171  4.41344481  0.90054367  3.87351200 
        x.8        x.11        x.12        x.13        x.14        x.15 
-3.02513407  3.65081989  2.43486098  0.79481451 -3.96716140 -3.53509069 
       x.16        x.17        x.19 
-1.63817806  1.62429959 -4.72088944 
```

Наведу ще раз список коефіцієнтів для справжньої моделі, яка використовувалася для генерації даних.

```
[1] "Beta-values list:"
 [1]  2.26110490  0.00000000 -4.97172271  4.41650527  0.72021482  3.83971442
 [7]  0.00000000 -2.92841800  0.00000000  0.00000000  3.76419525  2.40631339
[13]  0.99081832 -3.94591790 -3.64146732 -1.71236717  1.45211214 -0.01610153
[19] -4.62816600  0.00000000
```

Загалом бачимо, що модель, яка мінімізує тестовий MSE правильно визначила всі 5 нульових коефіцієнтів з списку бет і виключила їх з моделі. Щодо решти коефіцієнтів, то значень оцінок є точними до десятих.

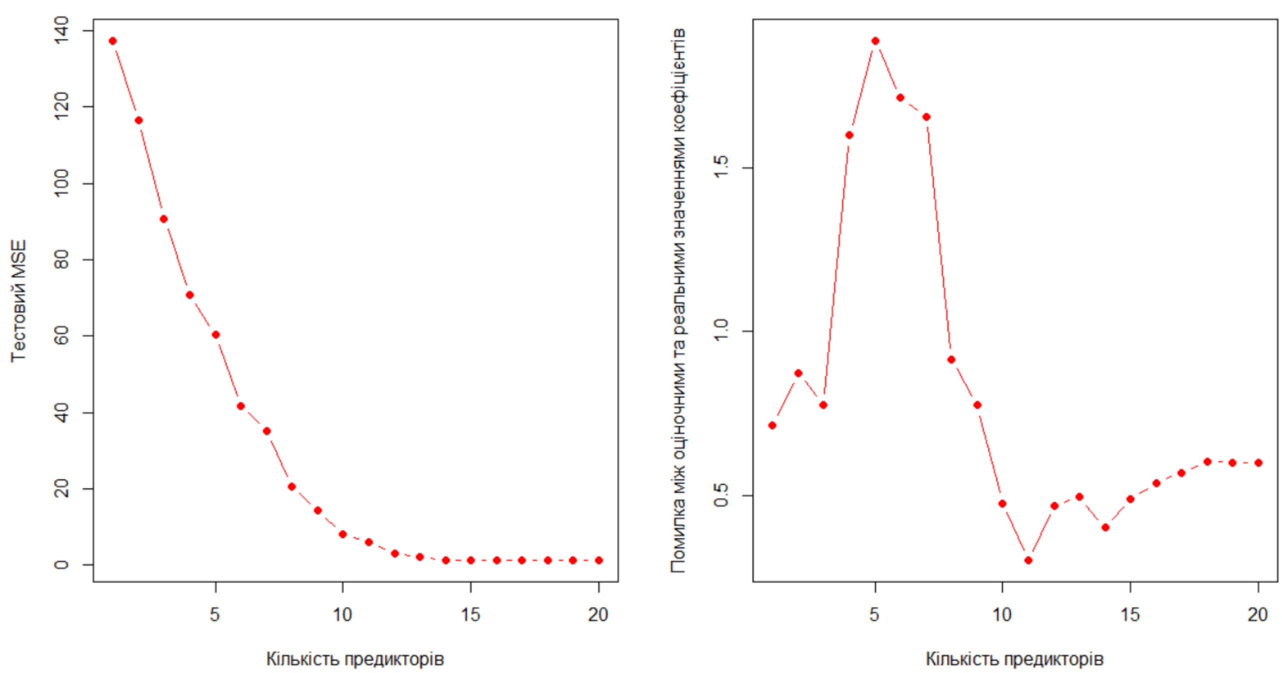
3.7 Побудуйте графік для відображення величини

$$\sqrt{\sum_{j=1}^{p}\left(\beta_j - \hat{\beta}_j^r\right)^2}$$

для всіх значень *r*, де $\hat{\beta}_j^r$ - оцінка *j*-ого коефіцієнта для найкращої моделі, що містить *r* коефіцієнтів. Прокоментуйте результати. Порівняйте отриманий графік з графіком тестового MSE з 3.4?

```
val.errors = rep(0, 20)
x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
```

```
for (i in 1:20) {
    coef_i = coef(reg.fit, id = i)
    val.errors[i] =
    sqrt(sum((betas[x_cols %in% names(coef_i)] -
         coef_i[names(coef_i) %in% x_cols])^2))
}

par(mfrow = c(1, 2))

BestSubsetSelection(x.test, y.test, "Тестовий MSE")

plot(val.errors, xlab = "Кількість предикторів",
 ylab = "Помилка між оціночними та реальними значеннями коефіцієнтів",
 col = "red", pch = 19, type = "b")
```



В результаті бачимо, що модель з 11-ма предикторами мінімізує помилку між оціночними та справжніми значеннями коефіцієнтів. Як було сказано раніше модель з 14-ма змінними є найкращою для тестового MSE. Проте варто сказати, що модель з 14-ма змінними є найкращою після моделі з 11-ма для помилки між оціночними та справжніми значеннями коефіцієнтів, але якоїсь явної кореляції між двома методами немає.

4. Спробуємо передбачити рівень злочинності на основі набору даних Boston.

```
      crim                zn             indus            chas        
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
 1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
 Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
 Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
      nox               rm             age              dis        
 Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
 1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
 Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
 Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
 3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
 Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
      rad              tax           ptratio          black       
 Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
 1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
 Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
 Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
 3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
 Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
     lstat            medv      
 Min.   : 1.73   Min.   : 5.00  
 1st Qu.: 6.95   1st Qu.:17.02  
 Median :11.36   Median :21.20  
 Mean   :12.65   Mean   :22.53  
 3rd Qu.:16.95   3rd Qu.:25.00  
 Max.   :37.97   Max.   :50.00  
```

Загальна характеристика даних Boston

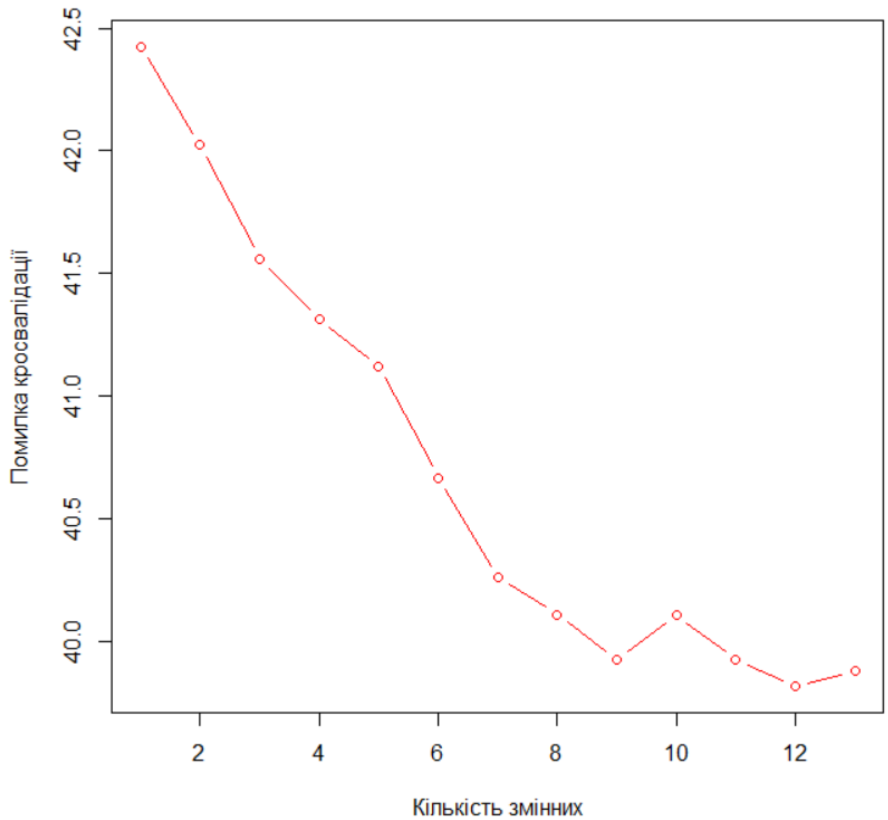
4.1-4.3 Застосуйте методи вибору моделі регресії, розглянуті раніше, такі як вибір найкращої підмножини, ласо, гребенева регресія та PCR. Представте та обговоріть результати щодо підходів, які ви використовуєте.

Запропонуйте модель, яка мала б добре працювати і обґрунтуйте свою відповідь. Для оцінки якості моделі використайте помилки валідаційної множини.

Чи включає обрана модель всі предиктори? Чому?

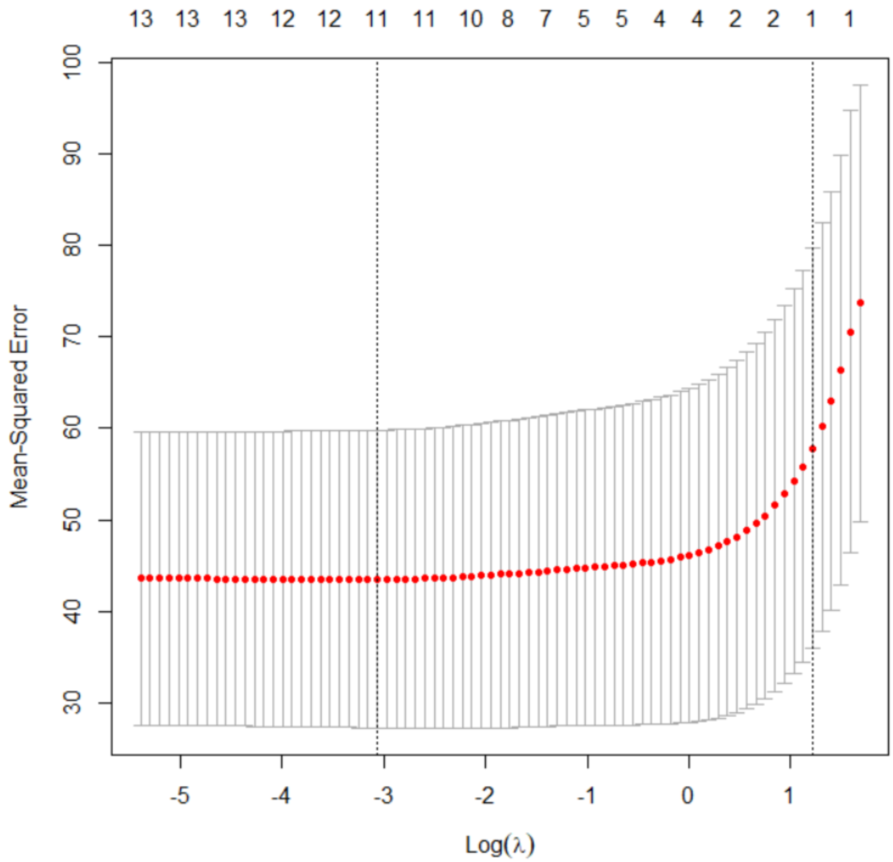
```
predict.regsubsets = function(object, newdata, id, ...) {
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coef_i = coef(object, id = id)
    xvars = names(coef_i)
    mat[, xvars] %*% coef_i
}

k = 10
folds = sample(1:k, nrow(Boston), replace = TRUE)
cv.errors = matrix(0, k, 13)

for (j in 1:k) {
    best.fit = regsubsets(crim ~ ., data = Boston[folds != j, ], nvmax = 13)
    for (i in 1:13) {
        pred = predict.regsubsets(best.fit, Boston[folds == j, ], id = i)
        cv.errors[j, i] = mean((Boston$crim[folds == j] - pred)^2)
    }
}

mean.cv.errors = rep(0, 13)
for (i in 1:13) {
  mean.cv.errors[i] = mean(cv.errors[, i])
}

cat("\n")
print(paste('Model size for min CV: ', which.min(mean.cv.errors),
 ', min CV error: ', min(mean.cv.errors)))
plot(mean.cv.errors, xlab = "Кількість змінних", ylab = "Помилка кросвалідації",
     col = "red", type = "b")
```

```
[1] "Model size for min CV:  12 , min CV error:  39.5171680324248"
```



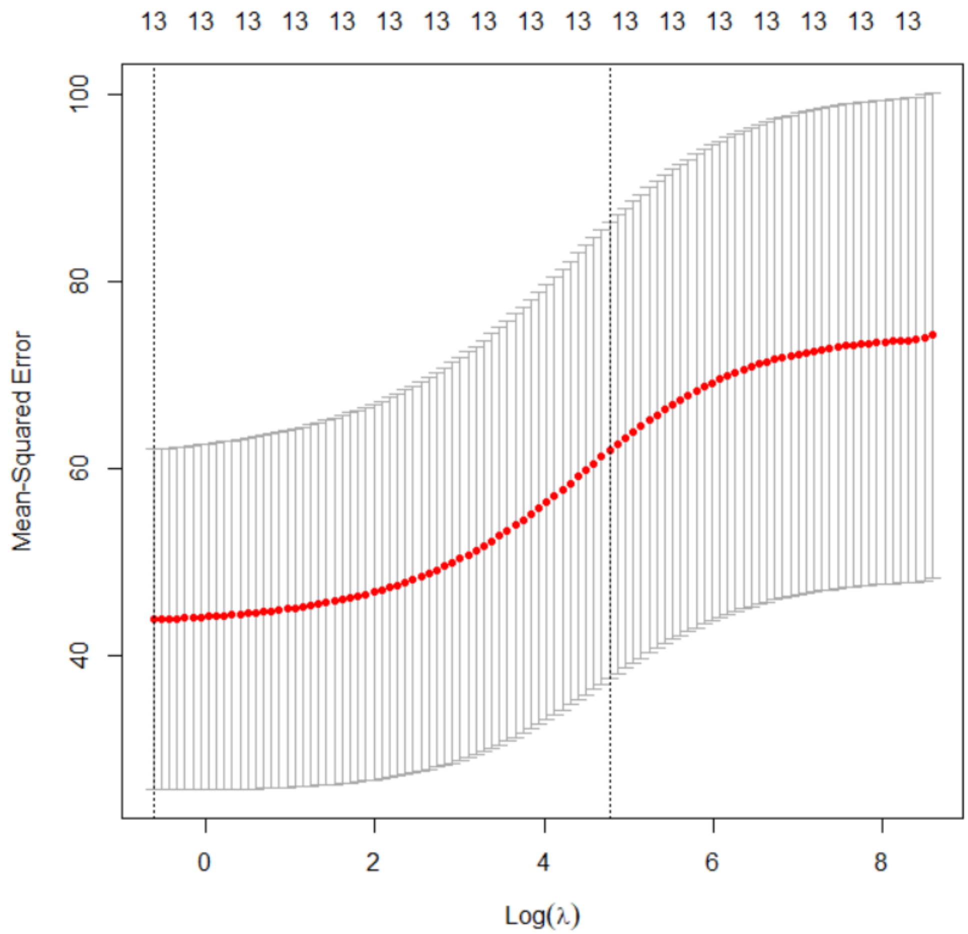

```
# lasso
cat("\n")
library(glmnet)
x = model.matrix(crim ~ ., Boston)[, -1]
y = Boston$crim

cv.lasso = cv.glmnet(x, y, alpha = 1, type.measure = "mse")
cat("\n")
print(paste('Lasso: Min lambda: ', cv.lasso$lambda.min,
 ', min CV error: ', min(cv.lasso$cvm)))
plot(cv.lasso)
```

```
[1] "Lasso: Min lambda:  0.0202364983610201 , min CV error:  42.3153137590084"
```



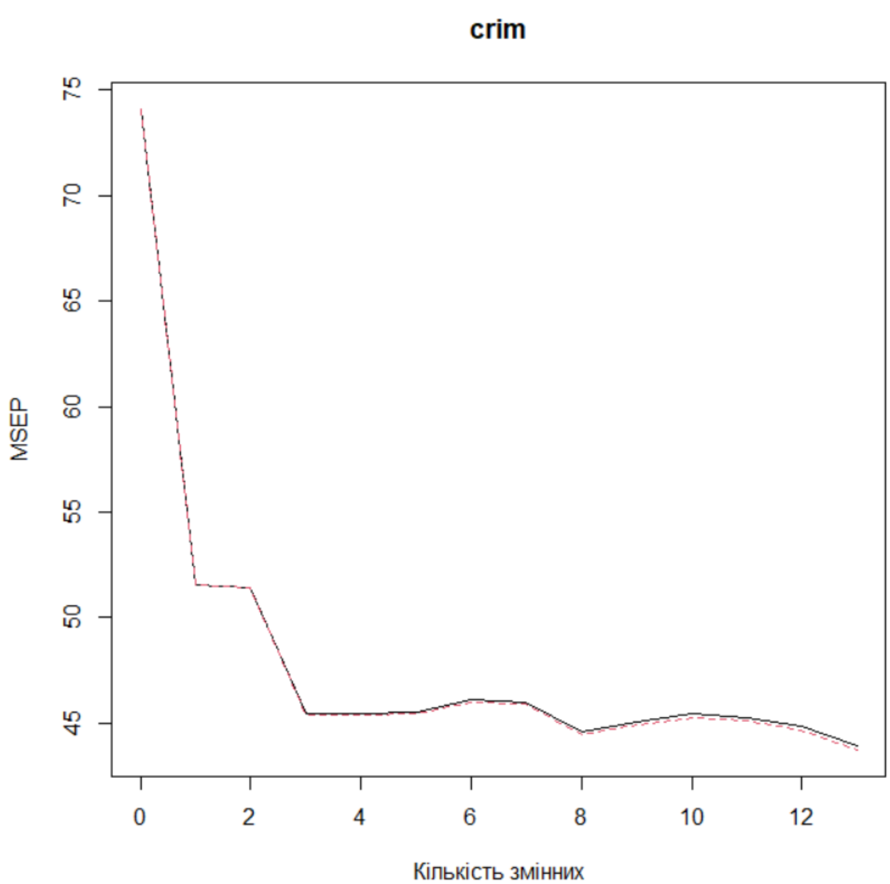

```
# ridge
cv.ridge = cv.glmnet(x, y, alpha = 0, type.measure = "mse")
cat("\n")
print(paste('Ridge: Min lambda: ', cv.ridge$lambda.min,
 ', min CV error: ', min(cv.ridge$cvm)))
plot(cv.ridge)
```

```
[1] "Ridge: Min lambda:  0.537499162479542 , min CV error:  42.8757612675674"
```




```
# PCR
cat("\n")
library(pls)

fit.pcr = pcr(crim ~ ., data = Boston, scale = TRUE, validation = "CV")
print(summary(fit.pcr))

cat("\n")
print(paste('Min M: ', which.min(fit.pcr$validation$adj),
 ', min CV error: ', min(fit.pcr$validation$adj)))

validationplot(fit.pcr, val.type = "MSEP", xlab = "Кількість змінних")
```

```
Data:   X dimension: 506 13
        Y dimension: 506 1
Fit method: svdpc
Number of components considered: 13

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV            8.61    7.227    7.227    6.786    6.771    6.787    6.814
adjCV         8.61    7.223    7.223    6.781    6.764    6.781    6.806
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       6.804    6.686    6.702     6.697     6.688     6.691     6.624
adjCV    6.796    6.677    6.693     6.687     6.678     6.677     6.609

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X       47.70    60.36    69.67    76.45    82.99    88.00    91.14    93.45
crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14    42.47
      9 comps  10 comps  11 comps  12 comps  13 comps
X       95.40     97.04     98.46     99.52     100.0
crim    42.55     42.78     43.04     44.13      45.4
NULL

[1] "Min M:  13 , min CV error:  40.5066541826557"
```



Оцінюючи результати помилки кросвалідації серед наведених вище методів можна сказати, що найнижчу помилку має метод найкращого вибору підмножини, а саме 39.7152, найвищу помилку має модель гребеневої регресії - 42.8758. При чому, варто наголосити, що модель з найнижчою помилкою для методу найкращого вибору підмножини має 12 предикторів.