Homework 6

Daniel Tshiani

2025-06-15

1

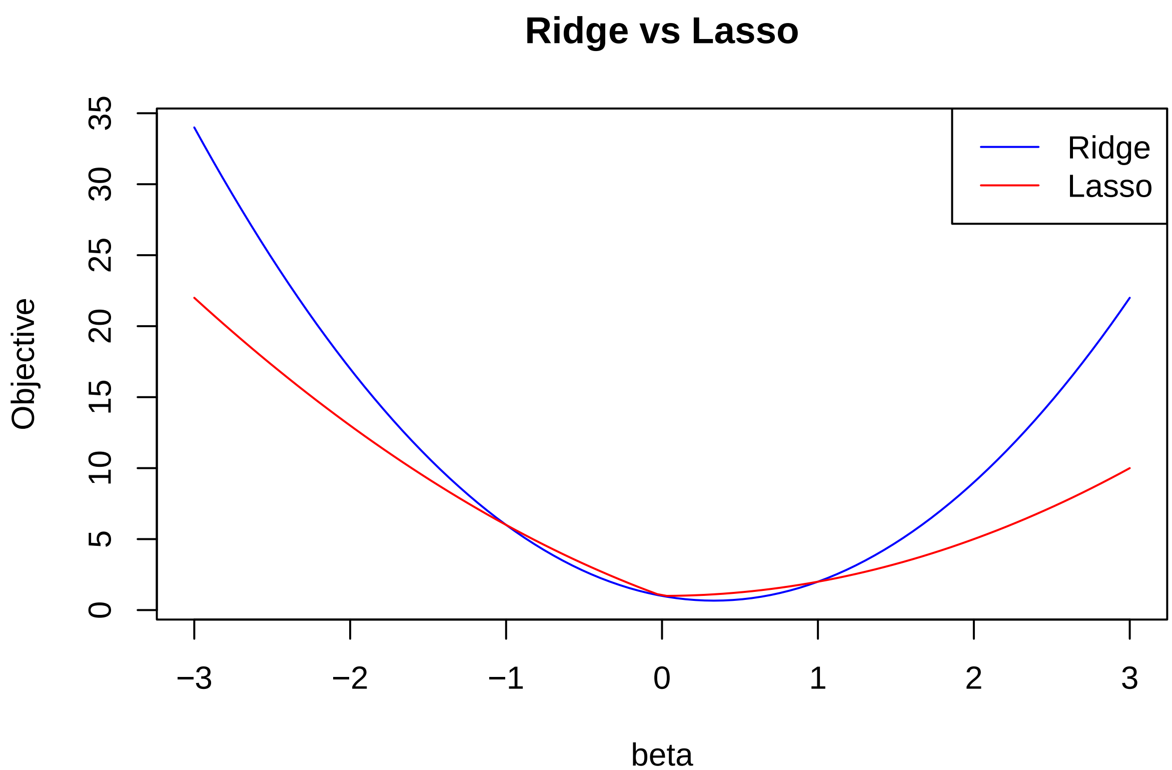
i. nonlinear
ii. nonlinear
iii. ridge regression
iv. lasso

2

a

```
Y <- 1
lambda <- 2
```

```
beta <- seq(-3, 3, length = 100)

RSS <- (Y - beta)^2
ridge_obj <- RSS + lambda * beta^2
lasso_obj <- RSS + lambda * abs(beta)

plot(beta, ridge_obj, type = "l", col = "blue", ylab = "Objective", main = "Ridge vs Lasso")
lines(beta, lasso_obj, col = "red")
legend("topright", legend = c("Ridge", "Lasso"), col = c("blue", "red"), lty = 1)
```




```r
beta_ridge <- Y / (1 + lambda)          # 1 / (1 + 2) = 0.333
beta_lasso <- if (Y > lambda / 2) Y - lambda / 2 else if (Y < -lambda / 2) Y + lambda / 2 else 0

beta_ridge
```

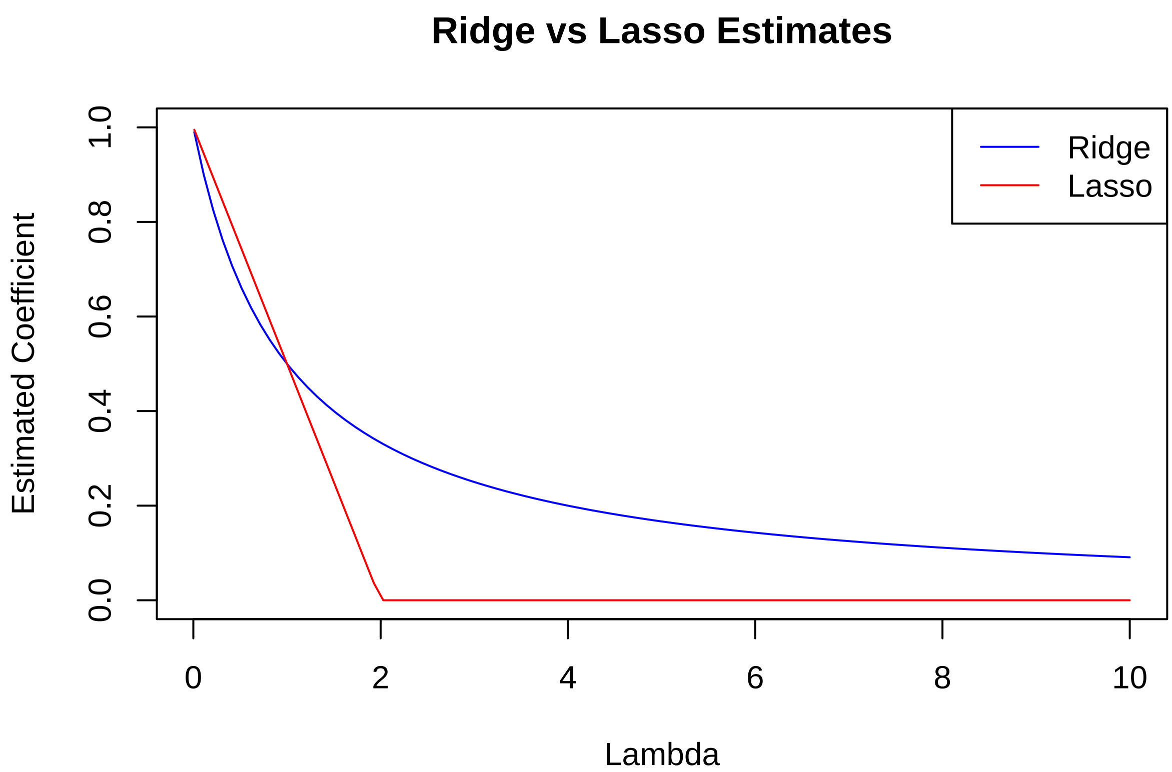
```
## [1] 0.3333333
```

```r
beta_lasso
```

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

b

```r
lambda_vals <- seq(0.01, 10, length = 100)
beta_ridge <- Y / (1 + lambda_vals)
beta_lasso <- ifelse(Y > lambda_vals / 2, Y - lambda_vals / 2,
                ifelse(Y < -lambda_vals / 2, Y + lambda_vals / 2, 0))

plot(lambda_vals, beta_ridge, type = "l", col = "blue", ylim = c(0,1),
     ylab = "Estimated Coefficient", xlab = "Lambda", main = "Ridge vs Lasso Estimates")
lines(lambda_vals, beta_lasso, col = "red")
legend("topright", legend = c("Ridge", "Lasso"), col = c("blue", "red"), lty = 1)
```



3

a

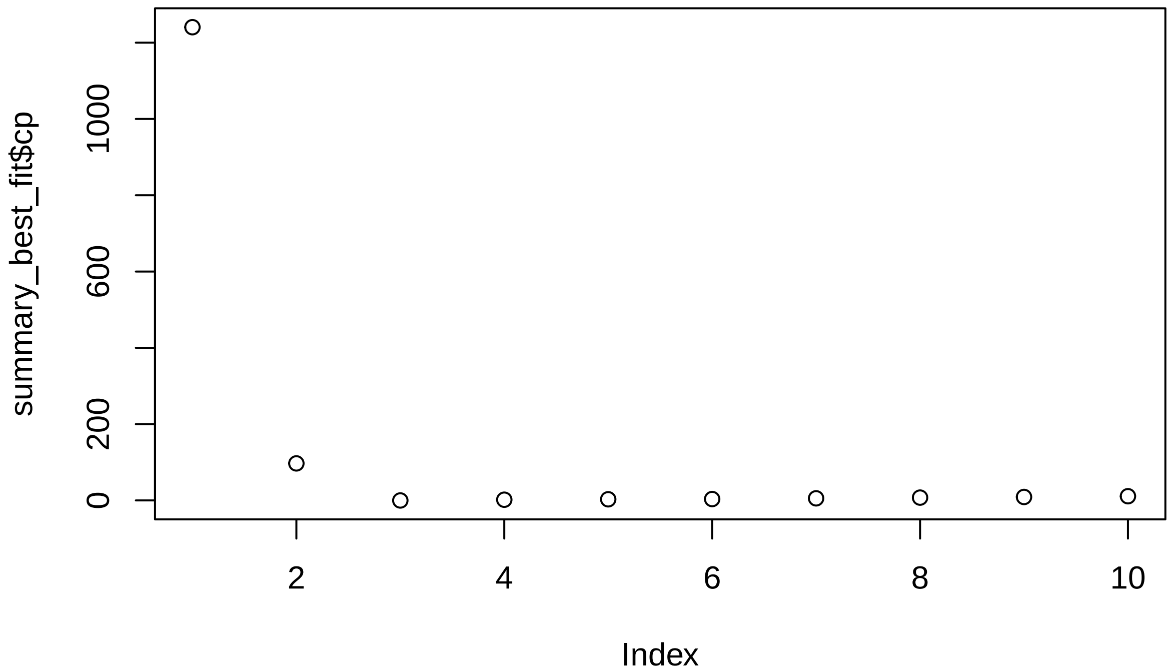
```r
set.seed(1234)
x <- rnorm(100)
error <- rnorm(100)
```

b

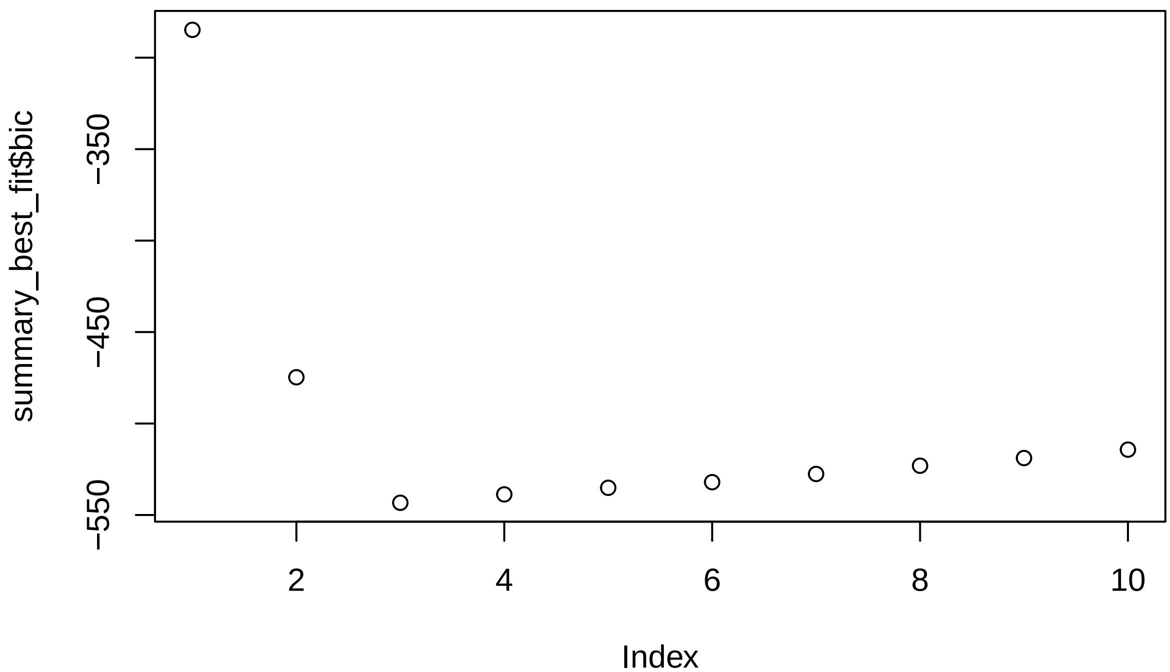
```r
beta0 <- 1
beta1 <- 2
beta2 <- 3
beta3 <- 4
Y <- beta0 + beta1*x + beta2*x^2 + beta3*x^3 + error
```

c

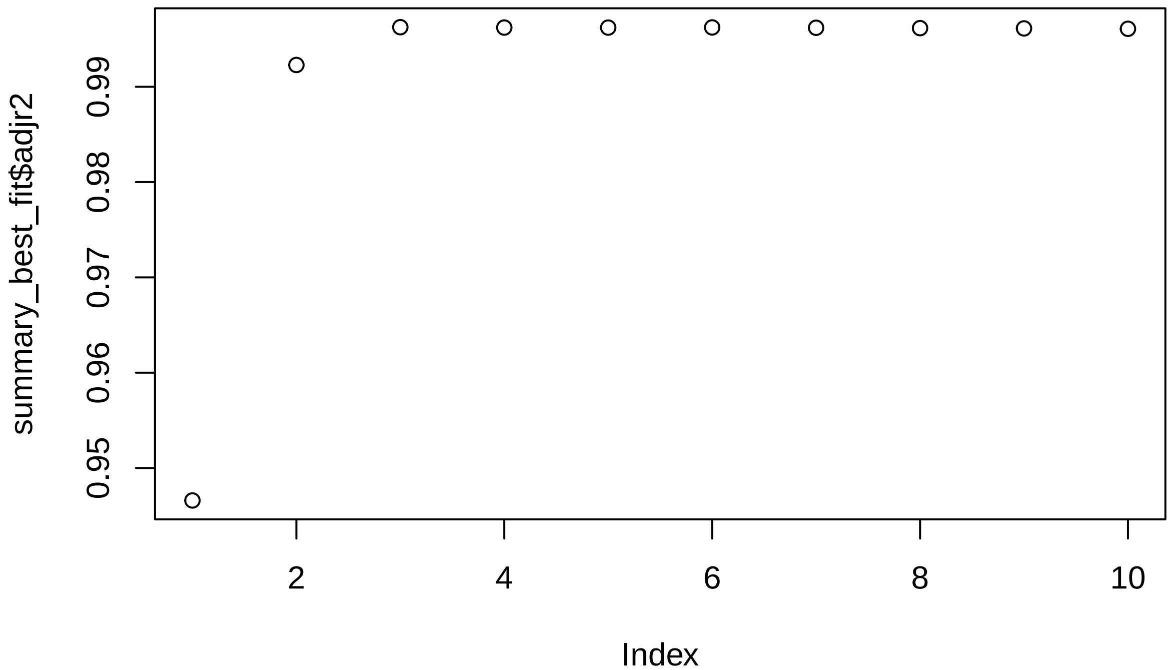
```r
library(leaps)

x_poly <- data.frame(poly(x,10, raw = T))
data <- data.frame(Y, x_poly)
best_fit <- regsubsets(Y ~., data = data, nvmax = 10)

summary_best_fit <- summary(best_fit)

plot(summary_best_fit$cp)
```




```r
plot(summary_best_fit$bic)
```



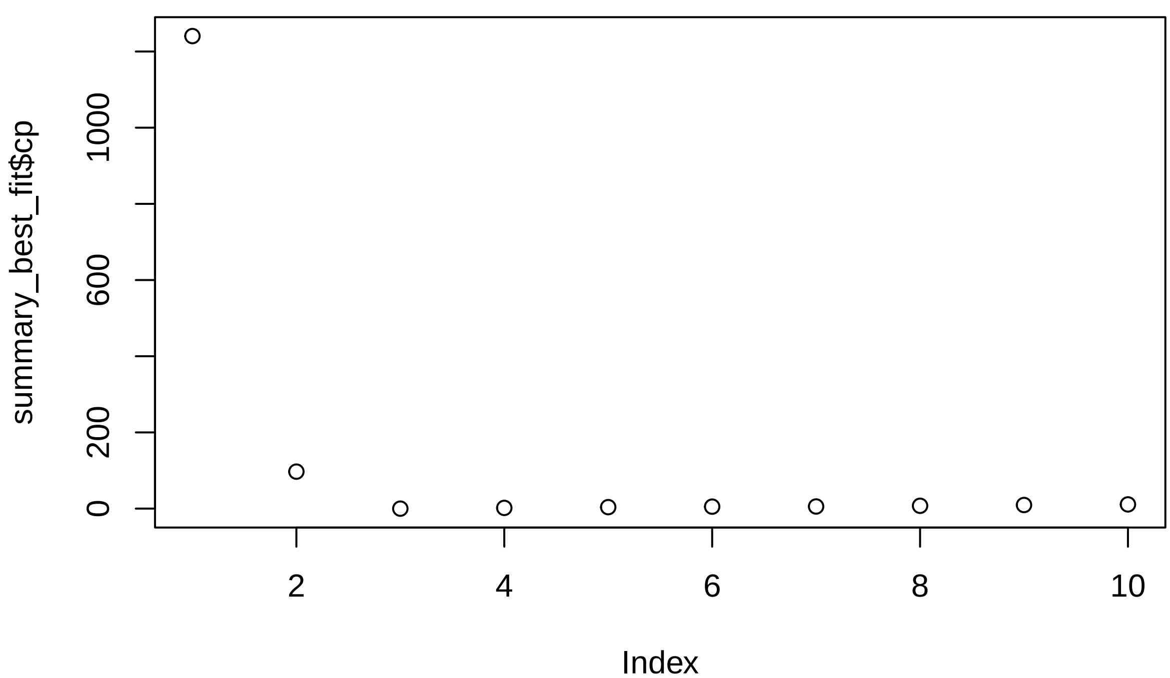

```r
plot(summary_best_fit$adjr2)
```



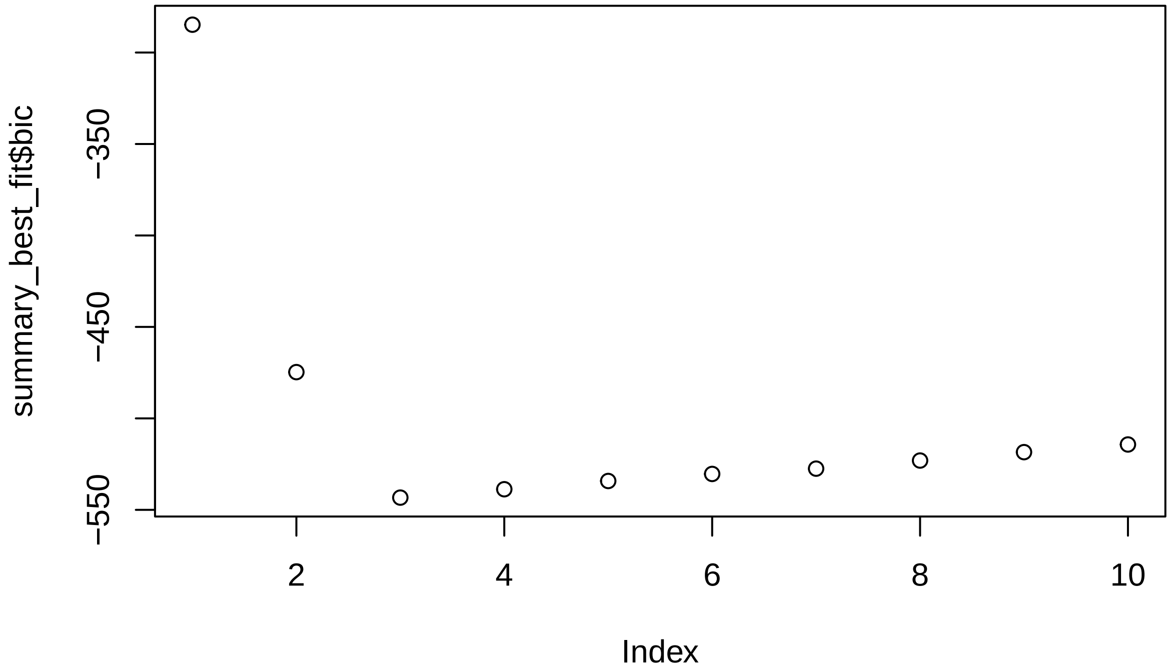
using CP, it looks like 3 predictors is the best. same for BIC. 3 predictors looks best. ADJR2 also suggest 3 predictos is the best. so the best model would be Y = beta0 + beta1 x1 + beta2 x1ˆ2 + beta3 x1ˆ3

d

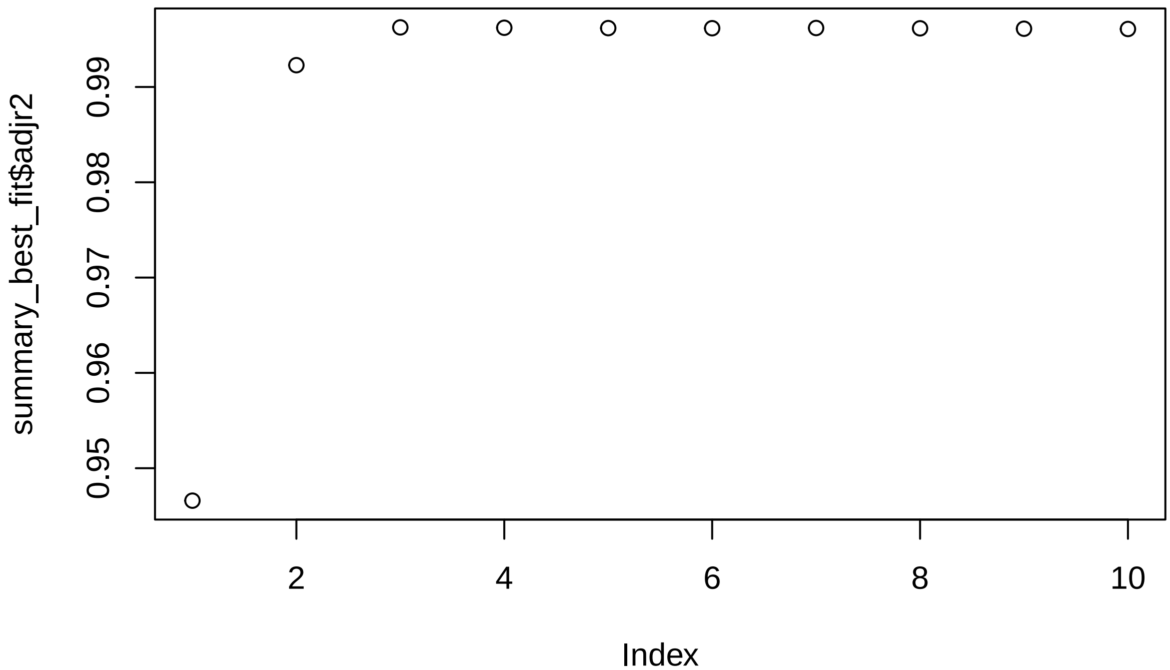
```
best_fit <- regsubsets(Y ~., data=data, nvmax = 10, method = "forward")

summary_best_fit <- summary(best_fit)

plot(summary_best_fit$cp)
```



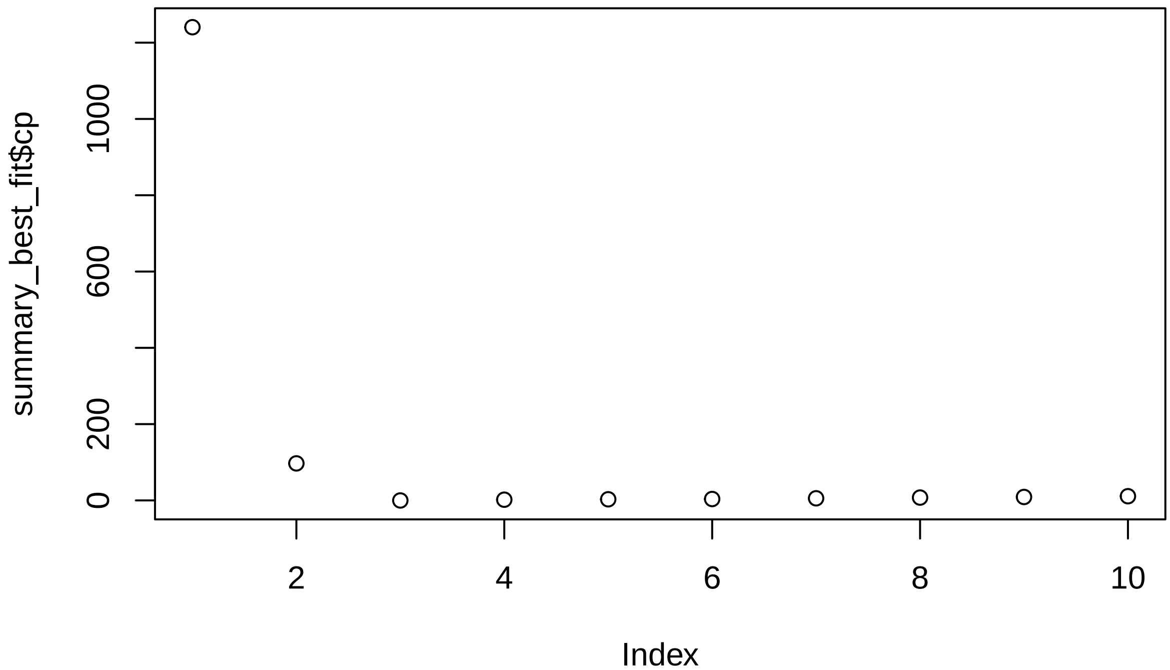
```r
plot(summary_best_fit$bic)
```



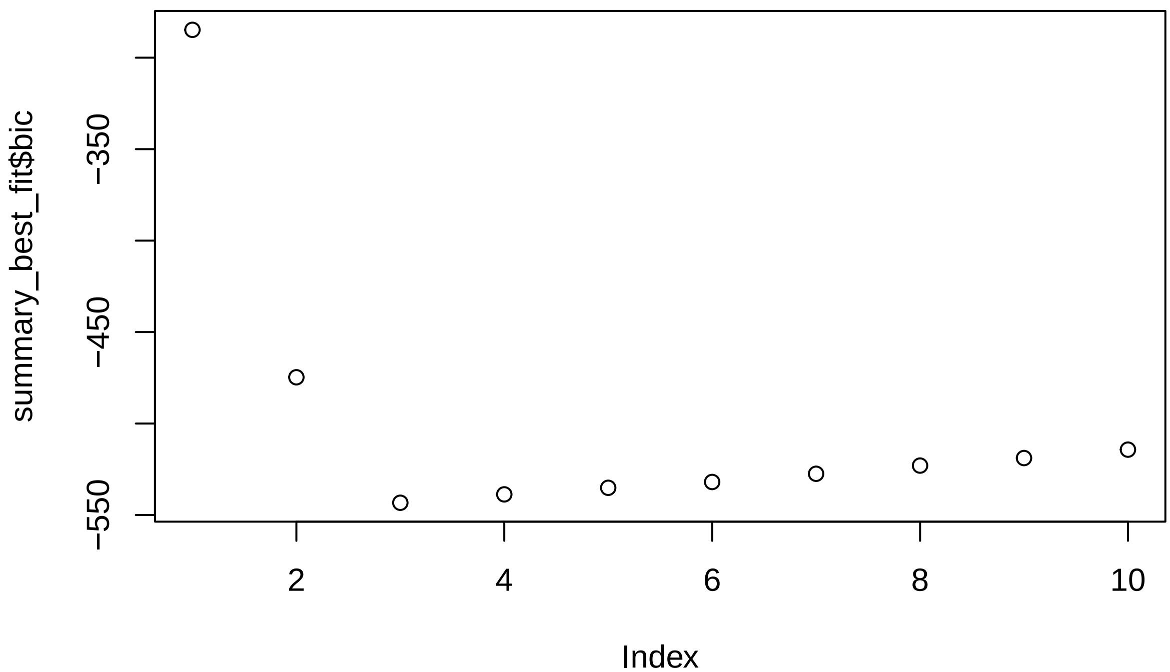

```r
plot(summary_best_fit$adjr2)
```



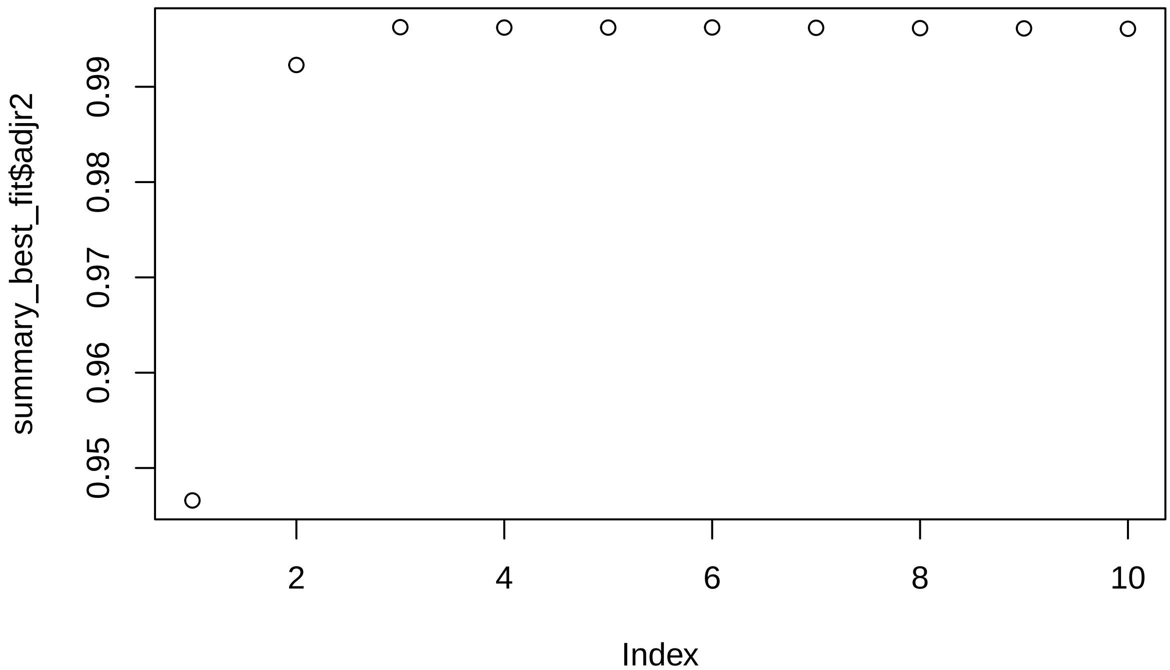

```r
best_fit <- regsubsets(Y ~., data=data, nvmax = 10, method = "backward")

summary_best_fit <- summary(best_fit)

plot(summary_best_fit$cp)
```




```
plot(summary_best_fit$bic)
```




```
plot(summary_best_fit$adjr2)
```



the forward and backward models suggest 3 predictors as well

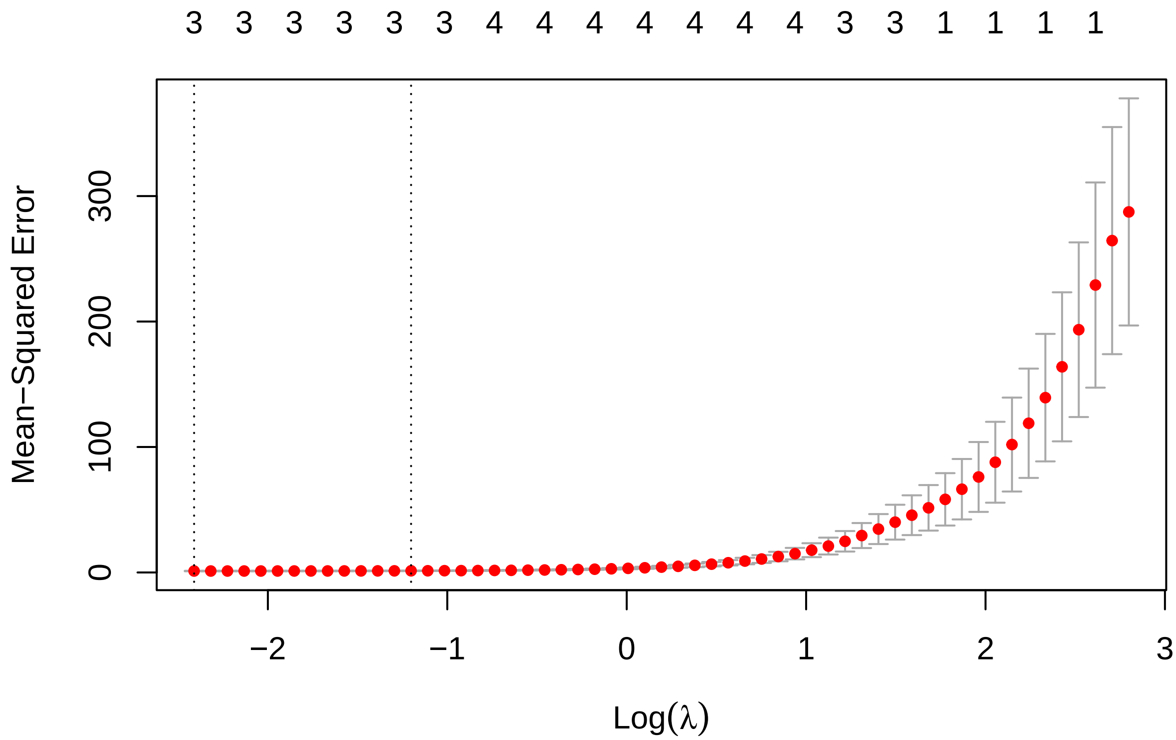
e

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-9
```

```
X_matrix <- model.matrix(Y ~ ., data = data)[, -1]
cv_lasso <- cv.glmnet(X_matrix, Y, alpha = 1)

plot(cv_lasso)
```



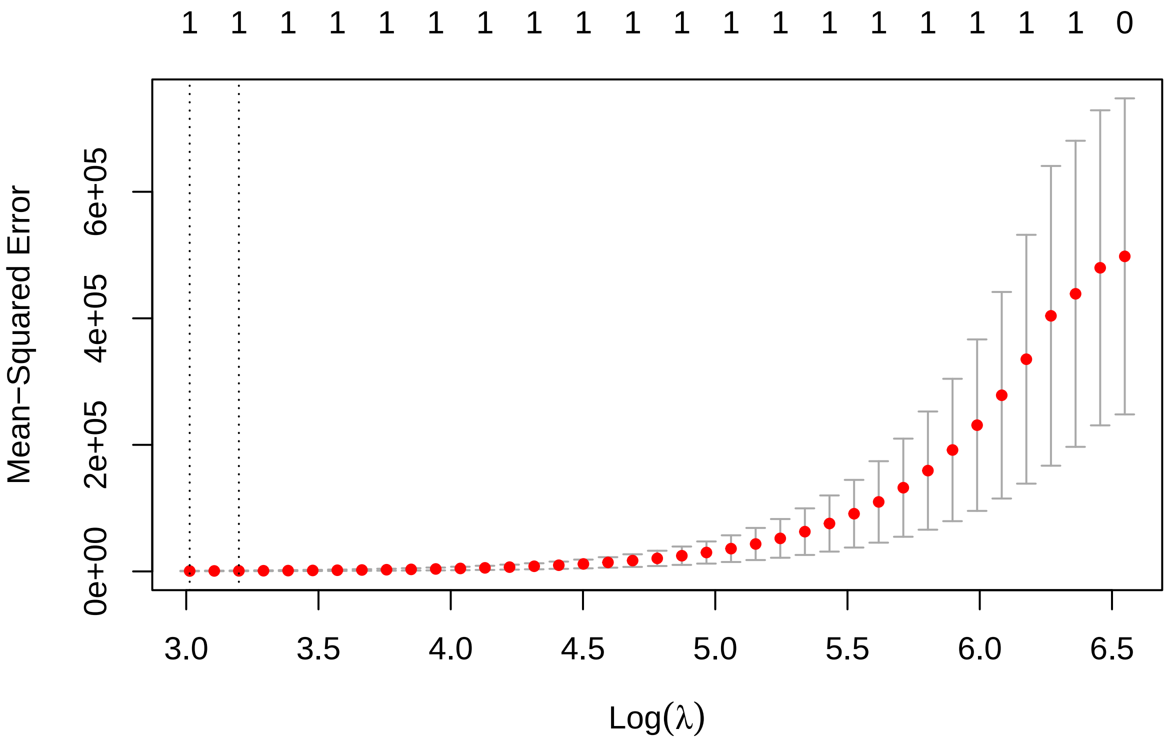

```
coef(cv_lasso, s = "lambda.min")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                    s0
## (Intercept) 1.178705
## X1          1.847133
## X2          2.838895
## X3          4.028284
## X4          .
## X5          .
## X6          .
## X7          .
## X8          .
## X9          .
## X10         .
```

x4 and above got eliminated

f

```
Y <- beta0 + 7 * x^7 + error

data2 <- data.frame(Y, poly(x, 10, raw = TRUE))
best_fit2 <- regsubsets(Y ~ ., data = data2, nvmax = 10)
summary_best2 <- summary(best_fit2)

X_matrix2 <- model.matrix(Y ~ ., data = data2)[, -1]
cv_lasso2 <- cv.glmnet(X_matrix2, Y, alpha = 1)
plot(cv_lasso2)
```




```r
coef(cv_lasso2, s = "lambda.min")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                   s0
## (Intercept) 2.952989
## X1          .
## X2          .
## X3          .
## X4          .
## X5          .
## X6          .
## X7          6.795857
## X8          .
## X9          .
## X10         .
```

4

a

```r
load("../data/College.rda")
library(ISLR)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
set.seed(1234)
train <- sample(1:nrow(College), nrow(College)/2)
test <- setdiff(1:nrow(College), train)

train_x <- model.matrix(Apps ~ ., data = College)[train, -1]
test_x <- model.matrix(Apps ~ ., data = College)[test, -1]
train_y <- College$Apps[train]
test_y <- College$Apps[test]
```

```r
lm_fit <- lm(Apps ~ ., data = College, subset = train)
pred_lm <- predict(lm_fit, newdata = College[test, ])
mean((pred_lm - test_y)^2)
```

```
## [1] 998713.2
```

b

```r
ridge_fit <- cv.glmnet(train_x, train_y, alpha = 0)
ridge_pred <- predict(ridge_fit, s = ridge_fit$lambda.min, newx = test_x)
mean((ridge_pred - test_y)^2)
```

```
## [1] 963730.7
```

c

```r
lasso_fit <- cv.glmnet(train_x, train_y, alpha = 1)
lasso_pred <- predict(lasso_fit, s = lasso_fit$lambda.min, newx = test_x)
mean((lasso_pred - test_y)^2)
```

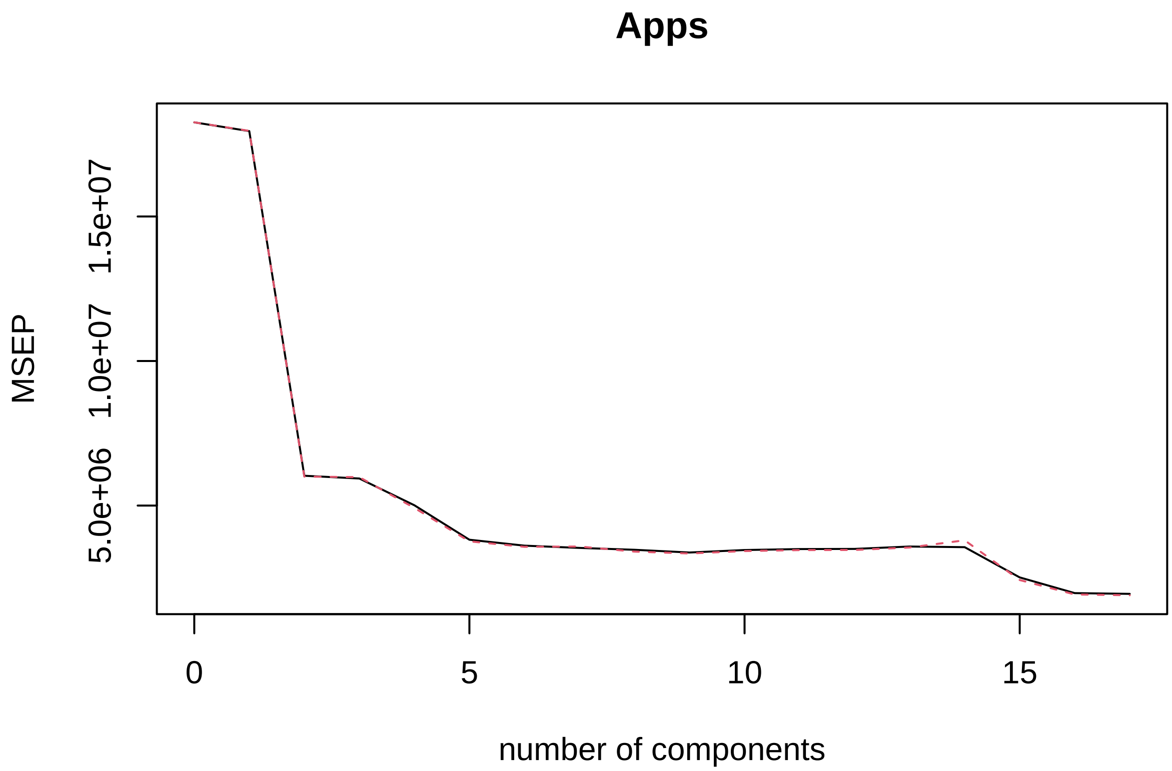
```
## [1] 965527.3
```

d

```r
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

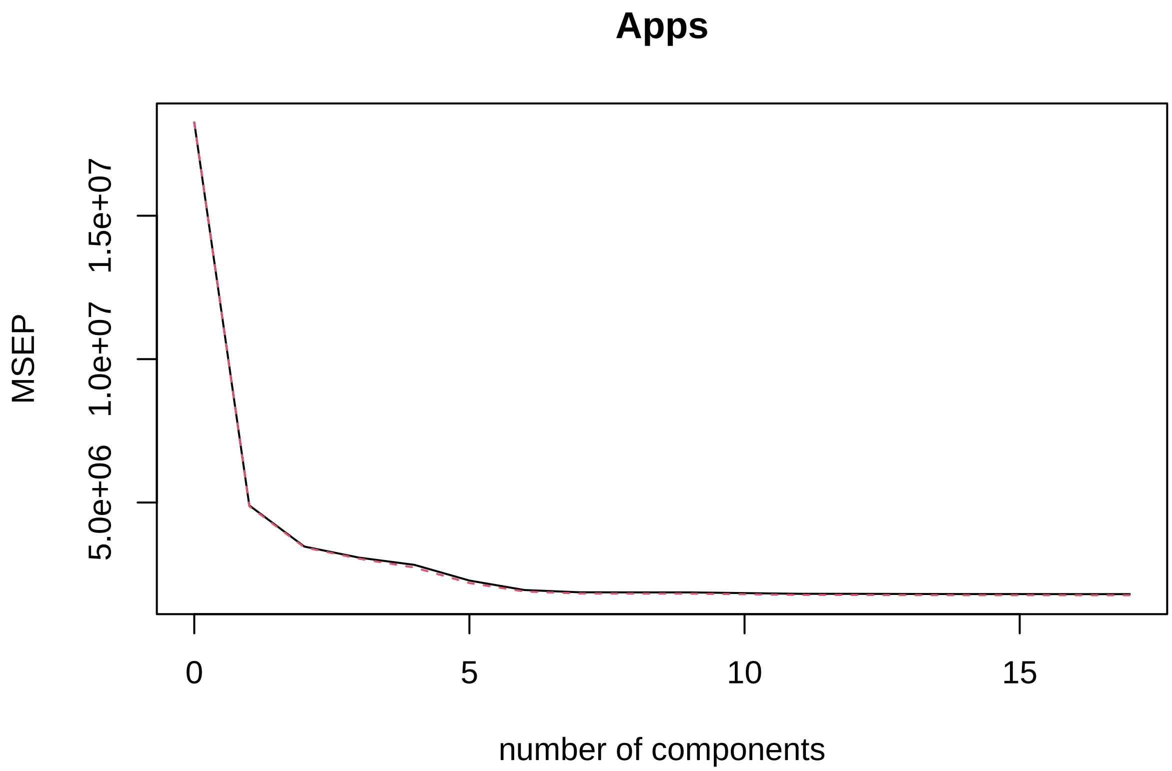
```r
pcr_fit <- pcr(Apps ~ ., data = College, subset = train, scale = TRUE, validation = "CV")
validationplot(pcr_fit, val.type = "MSEP")
```




```r
pcr_pred <- predict(pcr_fit, newdata = College[test, ], ncomp = 5)
mean((pcr_pred - test_y)^2)
```

```
## [1] 1734251
```

```r
pls_fit <- plsr(Apps ~ ., data = College, subset = train, scale = TRUE, validation = "CV")
validationplot(pls_fit, val.type = "MSEP")
```




```r
pls_pred <- predict(pls_fit, newdata = College[test, ], ncomp = 5)
mean((pls_pred - test_y)^2)
```

```
## [1] 1061050
```