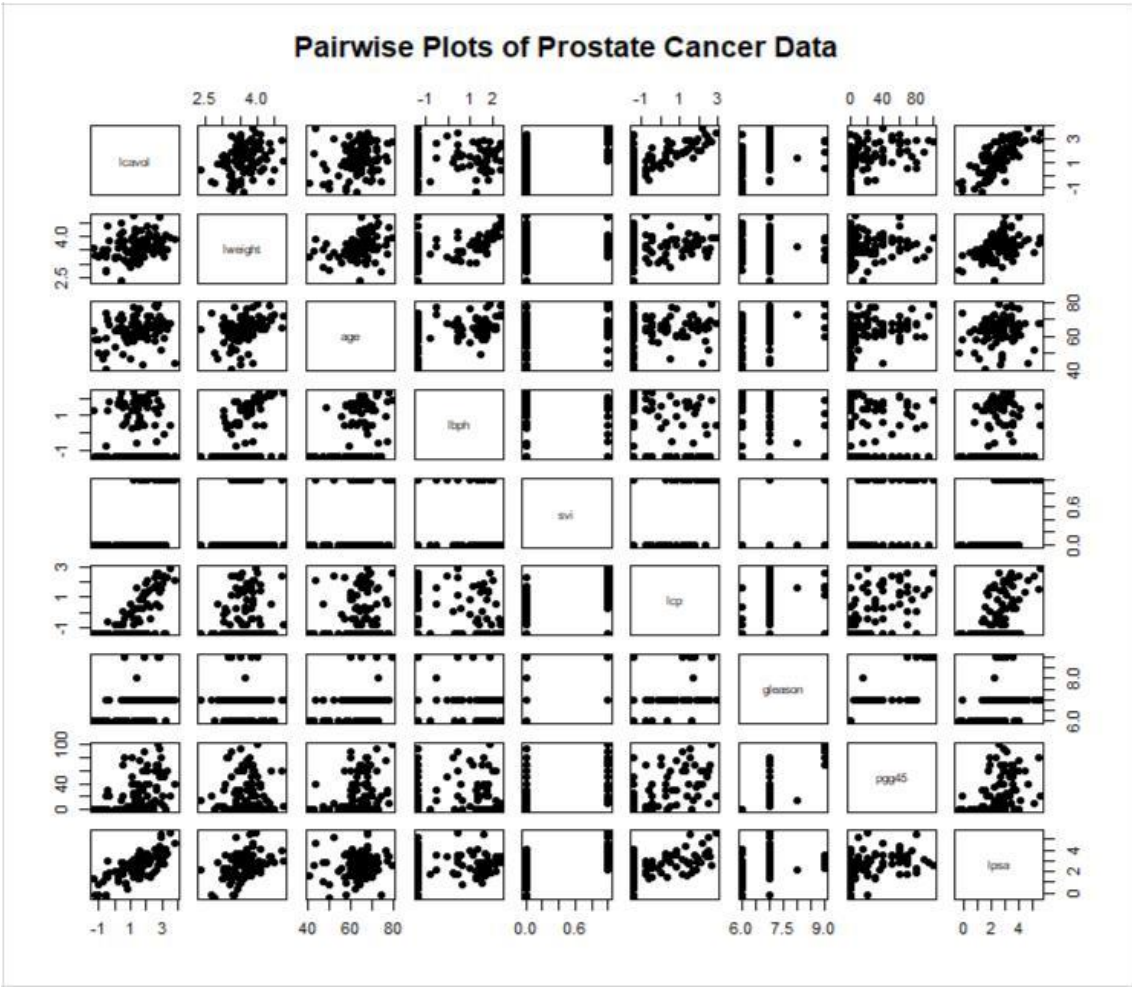
R code

```
# Read Prostate cancer data
library(ElemStatLearn)
?prostate
sum(is.na(prostate))
str(prostate)
View(prostate)
```

 # Exploratory analysis of the data:

```
prostate.new = prostate[, -10]
str(prostate.new)
head(prostate.new)

# Pairwise plots
pairs(prostate.new, main = "Pairwise Plots of Prostate Cancer Data", pch =
19)
```



From the pairwise plot, it could be seen that lcavol, lweight, lcp and pgg45 predictors appear to h ave a positive relationship in predicting lpsa. No specific patterns are visible in other predictors: age, lbph, svi and gleason.

From the correlation matrix, we can see that all the predictors have a positive correlation with lps a. lcavol has a strong correlation, while lweight, svi, lcp and pgg45 have moderate correlations, a nd age, lbph and gleason have weak correlations.

```
#.... Linear regression model .....#
  # We  first centered and scaled the explanatory variables to fit a usual linear regression model.

# Centering & scaling predictors
X = prostate.new[, -9]
  head(X)
  new.X = scale(X, center = TRUE, scale = TRUE)

  newdata = cbind(new.X, prostate.new$lpsa)
  colnames(newdata)[colnames(newdata) == ""] =
  "lpsa" newdata = as.data.frame(newdata)
  head(newdata)

  # Fit a full model
  model1 = lm(lpsa ~ ., data = newdata)
  summary(model1)
  anova(model1)

# 10-Fold CV test error rate
set.seed(12345)
  glm.fit = glm(lpsa ~ ., data = newdata)
  cv.error = cv.glm(newdata, glm.fit, K =
  10)$delta[1] cv.error

  library(DAAG)
  cv.lm(data = newdata, model1, m = 10)
```

 # Summary of fitting a linear model using usual least squares:

```
Call:
lm(formula = lpsa ~ ., data = newdata)

Residuals:
    Min      1Q  Median      3Q     Max
-1.7664 -0.3551 -0.0033  0.3809  1.5577

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.4784     0.0710   34.90  < 2e-16 ***
lcavol         0.6651     0.1035    6.43  6.5e-09 ***
lweight        0.2665     0.0861    3.10   0.0026 **
age           -0.1582     0.0825   -1.92   0.0585 .
```

```
lbph           0.1403     0.0840    1.67    0.0985 .
svi            0.3153     0.0998    3.16    0.0022 **
lcp           -0.1483     0.1257   -1.18    0.2412
gleason        0.0355     0.1122    0.32    0.7521
pgg45          0.1257     0.1231    1.02    0.3100
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.699 on 88 degrees of freedom
Multiple R-squared:  0.663,    Adjusted R-squared:  0.633
F-statistic: 21.7 on 8 and 88 DF,  p-value: <2e-16
```

The *p*-value associated with the F-statistic is essentially zero, which implies that at least one of the predictors must be related to lpsa. According to individual *p*-values for each variable, it seems that lcavol, lweight and svi are related to lpsa, but there is no evidence that other predictors are associated with lpsa, in the presence of those three.

Using 10-fold cross-validation, the test error rate = 0.579.

```
#.. Best-subset selection .....#
```

R code:

```
library(leaps)
totpred = 8
model2 = regsubsets(lpsa ~ ., data = newdata, nvmax = totpred)
model2.summary = summary(model2)
model2.summary
model2.summary$cp
which.min(model2.summary$cp)
model2.summary$bic
which.min(model2.summary$bic)
coef(model2, 3)

# 10-Fold CV test error rate
# A function to get predictions for a model from a regsubsets
object predict.regsubsets = function(object, new.data, id, ...) {
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, new.data)
  coefi = coef(object, id = id) xvars
  = names(coefi)
  mat[, xvars] %*% coefi
}

K = 10
set.seed(12345)
folds = sample(rep(1:K, length = nrow(newdata)))
cv.errors = matrix(NA, K, totpred)

for (j in 1:K) {
  best.fit = regsubsets(lpsa ~ ., data = newdata[folds != j, ], nvmax =
totpred)
```

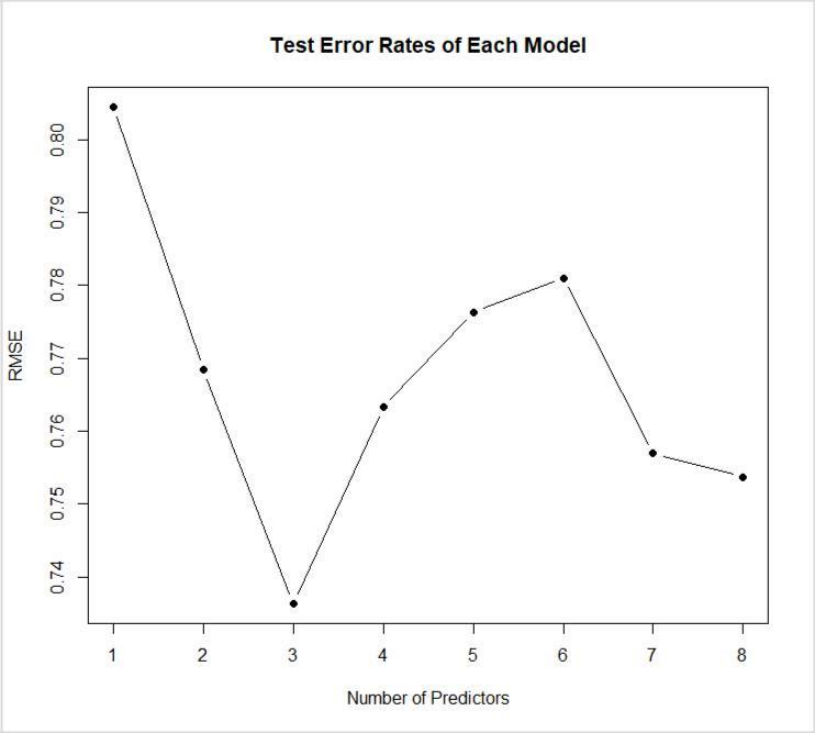
```
  for (i in 1:totpred) {
    pred = predict(best.fit, newdata[folds == j, ], id = i)
    cv.errors[j, i] = mean((newdata$lpsa[folds == j] - pred)^2)
  }
}
rmse.cv = sqrt(apply(cv.errors, 2, mean))
plot(rmse.cv, pch = 19, type = "b", main = "Test Error Rates of Each Model",
xlab = "Number of Predictors",
     ylab = "RMSE")
rmse.cv
which.min(rmse.cv)
coef(model2, 3)
```

Model summary using best-subset selection:

```
          lcavol lweight age lbph svi lcp gleason pgg45
1  ( 1 ) "*"    " "     " " " "  " " " " " "     " "
2  ( 1 ) "*"    "*"     " " " "  " " " " " "     " "
3  ( 1 ) "*"    "*"     " " " "  "*" " " " "     " "
4  ( 1 ) "*"    "*"     " " "*"  "*" " " " "     " "
5  ( 1 ) "*"    "*"     "*" "*"  "*" " " " "     " "
6  ( 1 ) "*"    "*"     "*" "*"  "*" " " " "     "*"
7  ( 1 ) "*"    "*"     "*" "*"  "*" "*" " "     "*"
8  ( 1 ) "*"    "*"     "*" "*"  "*" "*" "*"     "*"
```

If we use BIC statistic to select the best model, then the model with the lowest BIC = -79.7 is the three-variable model that contains only lcavol, lweight and svi.

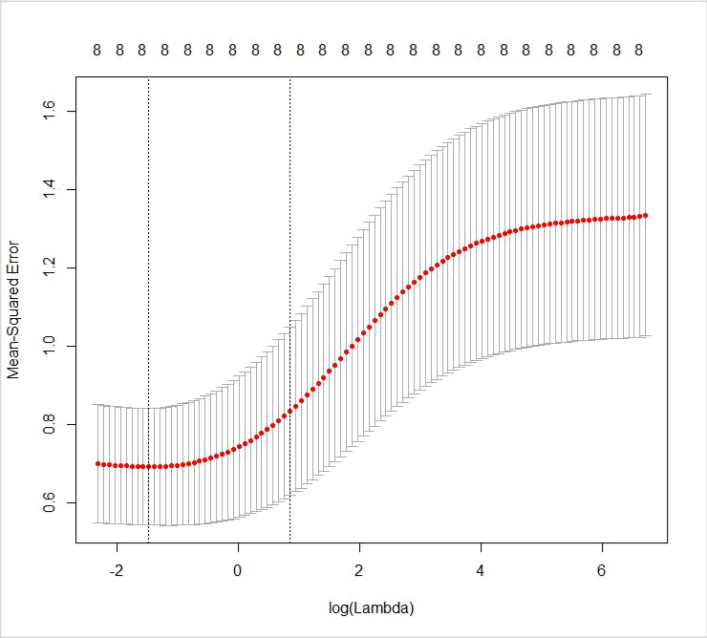
Using 10-fold cross-validation, we calculated test error rates of each model. The following figure shows that the model with three predictors has the lowest test error rate = 0.736.



```
#.... Ridge regression .....#

library(glmnet)

y = prostate.new$lpsa
x = model.matrix(lpsa ~ ., data = prostate.new)[, -1]

# Create training and test sets
set.seed(123)
train = sample(1:nrow(x), nrow(x) / 2)
test = (-train)
y.test = y[test]

# Use CV to choose the tuning parameter
set.seed(123)
cv.out = cv.glmnet(x[train, ], y[train], alpha = 0, nfolds =
10) plot(cv.out)
bestlam = cv.out$lambda.min
bestlam

# Estimate test error rate
grid = 10 ^ seq(10, -2, length = 100)
ridge.mod = glmnet(x[train, ], y[train], alpha = 0, lambda = grid, thresh =
1e-12)
ridge.pred = predict(ridge.mod, s = bestlam, newx = x[test, ])
mean((ridge.pred - y.test) ^ 2)

# Refit the model on the full dataset
out = glmnet(x, y, alpha = 0)

# Estimates for the best value of lambda
predict(out, type = "coefficients", s = bestlam)[1:9, ]
```

Fitting a model using ridge regression:

To choose the best tuning parameter , we use estimated test error rates based on 10-fold cross-validation. The value of that results in the smallest cross-validation error is 0.228 (see the foll owing figure) and the test MSE associated with this value of is 0.455.

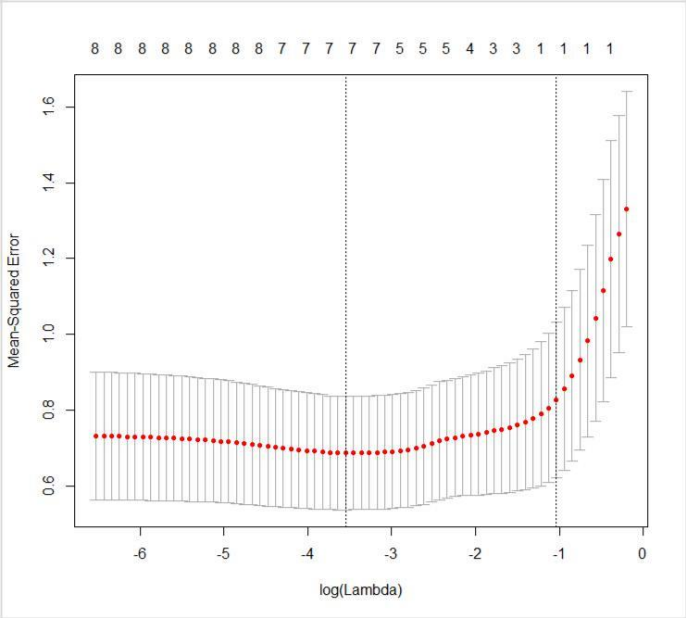


Then, we refit our ridge regression model on the full data set, using the value of chosen by cro ss-validation, and examine the coefficient estimates in the table at the end.

```
#.....using  LASSO .....#

# Use CV to choose the tuning parameter
set.seed(123)
   cv.out = cv.glmnet(x[train, ], y[train], alpha = 1, nfolds =
   10) plot(cv.out)
   bestlam =
   cv.out$lambda.min
   bestlam

# Estimate test error rate
   lasso.mod = glmnet(x[train, ], y[train], alpha = 1, lambda = grid)
   lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test, ])
   mean((lasso.pred - y.test)^2)

   # Refit the model on the full dataset
   out = glmnet(x, y, alpha = 1, lambda = grid)

   # Estimates for the best value of lambda
   lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:9, ]
   lasso.coef
   lasso.coef[lasso.coef != 0]
```

Fit a model using LASSO:

To choose the best tuning parameter , we again use estimated test error rates based on 10-fold cross-validation. The value of that results in the smallest cross-validation error is 0.0228 (see t he following figure) and the test MSE associated with this value of is 0.449.



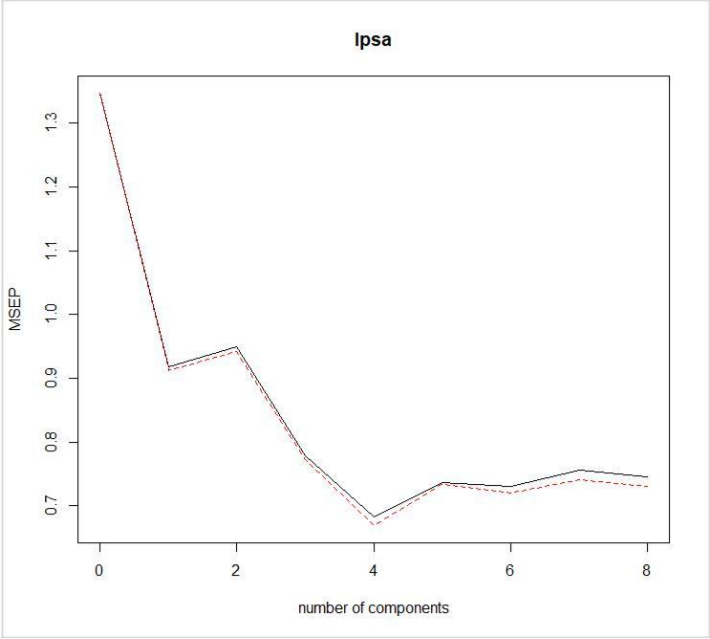
Then, we refit our LASSO model on the full data set, using the value of chosen by cross-valid ation, and examine the coefficient estimates (see part (h)). From the results, we see that the coeffi cient estimate of lcp is exactly zero, so the LASSO model with chosen by cross-validation co ntains only seven variables. For the coefficients of those variables see table at the end below.

```
#..... using PCR .....#

library(pls)

# Fit model to the training set
set.seed(123)
   pcr.fit = pcr(lpsa ~ ., data = prostate.new, subset = train, scale = TRUE,
   validation = "CV",
                segments = 10)
      validationplot(pcr.fit, val.type =
                     "MSEP")
# We see that lowest CV occurs when M = 4

# Compute test MSE for M = 4
   pcr.pred = predict(pcr.fit, x[test, ], ncomp = 4)
   mean((pcr.pred - y.test) ^ 2)
```

```
# Refit the model on the full dataset
pcr.fit = pcr(lpsa ~ ., data = prostate.new, scale = TRUE, ncomp = 4)
summary(pcr.fit)
coef(pcr.fit, ncomp = 4, intercept = TRUE)
```

Fitting a model using PCR:

Compute 10-fold cross-validation error for each possible value of *M*, the number of principle co mponents used.



We see that the smallest error rate occurs when *M* = 4 components are used, and the test MSE ass ociated with this value is 0.492. Then, we refit our PCR model on the full data set, using 4 comp onents, and examine the coefficient estimates (see part (h)).

```
#..... using PLS.......... #

# Fit model to the
training set set.seed(123)
pls.fit = plsr(lpsa ~ ., data = prostate.new, subset = train, scale =
TRUE, validation = "CV",
               segments = 10)
validationplot(pls.fit, val.type =
"MSEP") # We see that lowest CV
occurs when M = 2

# Compute test MSE for M = 2
pls.pred = predict(pls.fit, x[test, ], ncomp
= 2) mean((pls.pred - y.test)^2)
```

```
# Refit the model on the full dataset
pls.fit = plsr(lpsa ~ ., data = prostate.new, scale = TRUE, ncomp = 2)
summary(pls.fit)
coef(pls.fit, ncomp = 2, intercept = TRUE)
```

Fitting a model using PLS:

The lowest cross-validation error occurs when only $M = 2$ partial least squares directions are use d. Then we evaluate the corresponding test MSE which is 0.465. Then, we refit our PLS model o n the full data set, using 2 components, and examine the coefficient estimates (see part (h)).

 # Summary of the parameter estimates and test error rates:

Term	LS	Best Subset	Ridge	LASSO	PCR	PLS
Intercept	2.4784	2.478	-0.08275	0.17218	0.1844	-0.3005
lcavol	0.6651	0.620	0.41623	0.50937	0.2933	0.4762
lweight	0.2665	0.284	0.56686	0.56032	0.3001	0.3253
age	-0.1582		-0.01182	-0.01050	-0.0691	-0.0770
lbph	0.1403		0.07420	0.06815	0.1174	0.1203
svi	0.3153	0.276	0.61004	0.60007	0.2905	0.2707
lcp	-0.1483		0.01610		0.2772	0.1350
gleason	0.0355		0.07278	0.00895	-0.0169	-0.0010
pgg45	0.1257		0.00282	0.00240	0.0498	0.0173
Test Error	**0.579**	**0.736**	**0.455**	**0.449**	**0.492**	**0.465**

The smallest cross-validation test error rate is 0.449 from the linear model fitted using LASSO, a nd it contains only seven predictors. Therefore, we select linear model from LASSO as a good fit for the Prostate cancer data.