Question 20: Problem 9 in Section 6.8

Julia Lee

10/28/2018

A

```
data(College)
set.seed(11)
train_College <- College %>%
  sample_frac(0.5)

test_College <- College %>%
  setdiff(train_College)
```

B

```
mod <- lm(Apps ~., data = train_College)
pred_lm <- predict(mod, test_College)
lm_mse<-mean((pred_lm - test_College$Apps)^2)
lm_mse
```

```
## [1] 1538442
```

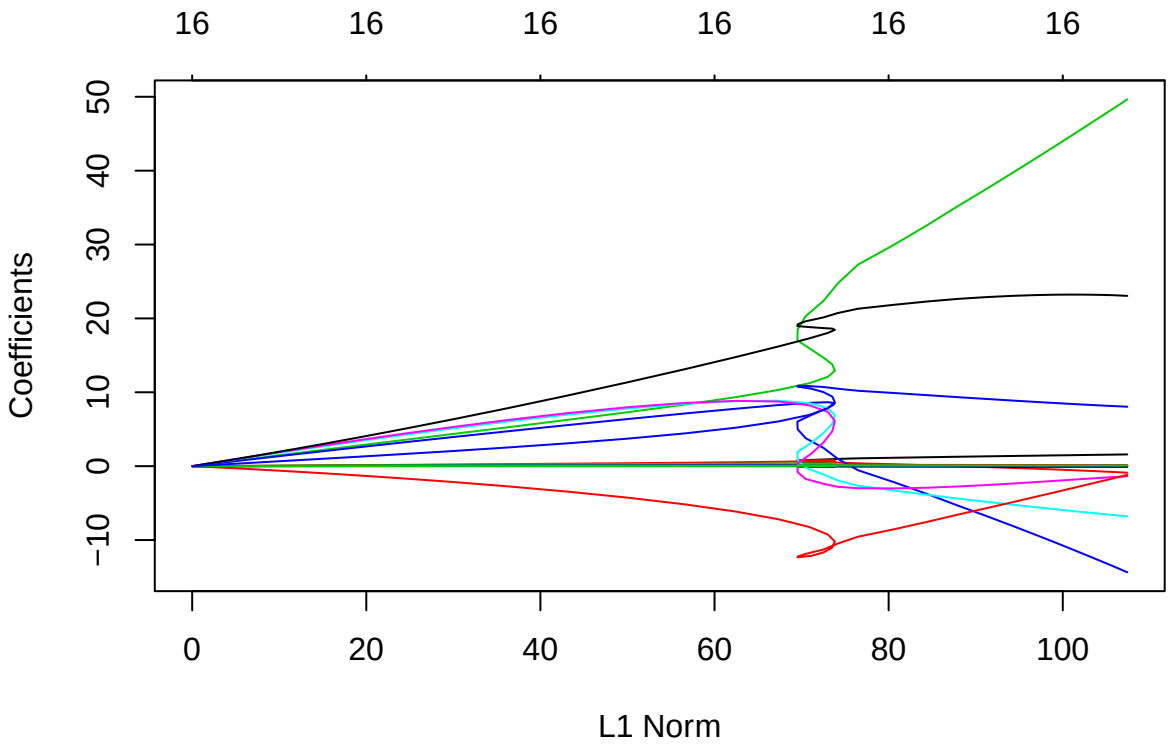
the test MSE is 1538442

C

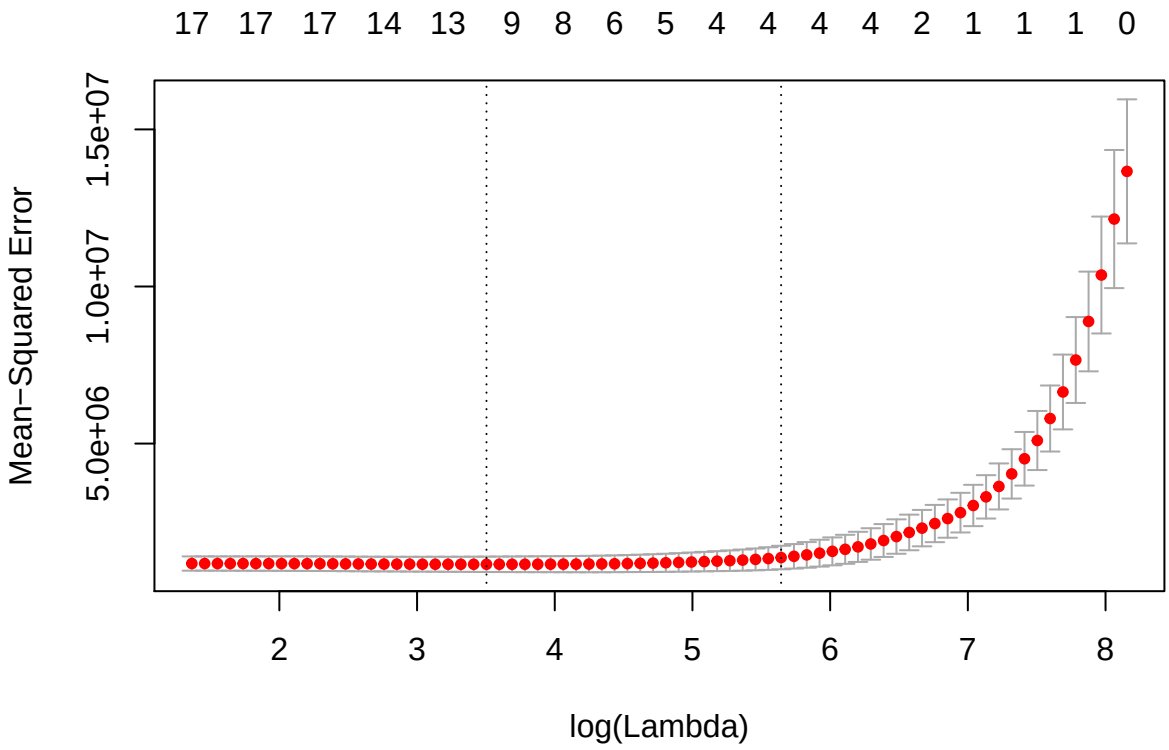
```
x <- model.matrix(Apps~., College)[,-2]
# leaving only the predictors
y <- College %>%
  dplyr::select(Apps) %>%
  unlist() %>%
  as.numeric()
grid <- 10^seq(10, -2, length = 100)
ridge_mod <- glmnet(x, y, alpha = 0, lambda = grid)
dim(coef(ridge_mod))
```

```
## [1]  18 100
```

```
plot(ridge_mod)
```




```
x_train <- model.matrix(Apps~., train_College)[,-1]
x_test <- model.matrix(Apps~., test_College)[,-1]
y_train = train_College %>%
  dplyr::select(Apps) %>%
  unlist() %>%
  as.numeric()
y_test = test_College %>%
  dplyr::select(Apps) %>%
  unlist() %>%
  as.numeric()
ridge_mod <- glmnet(x_train, y_train, alpha=0, lambda = grid, thresh = 1e-12)
cv.out = cv.glmnet(x_train, y_train, alpha = 1)
plot(cv.out)
```



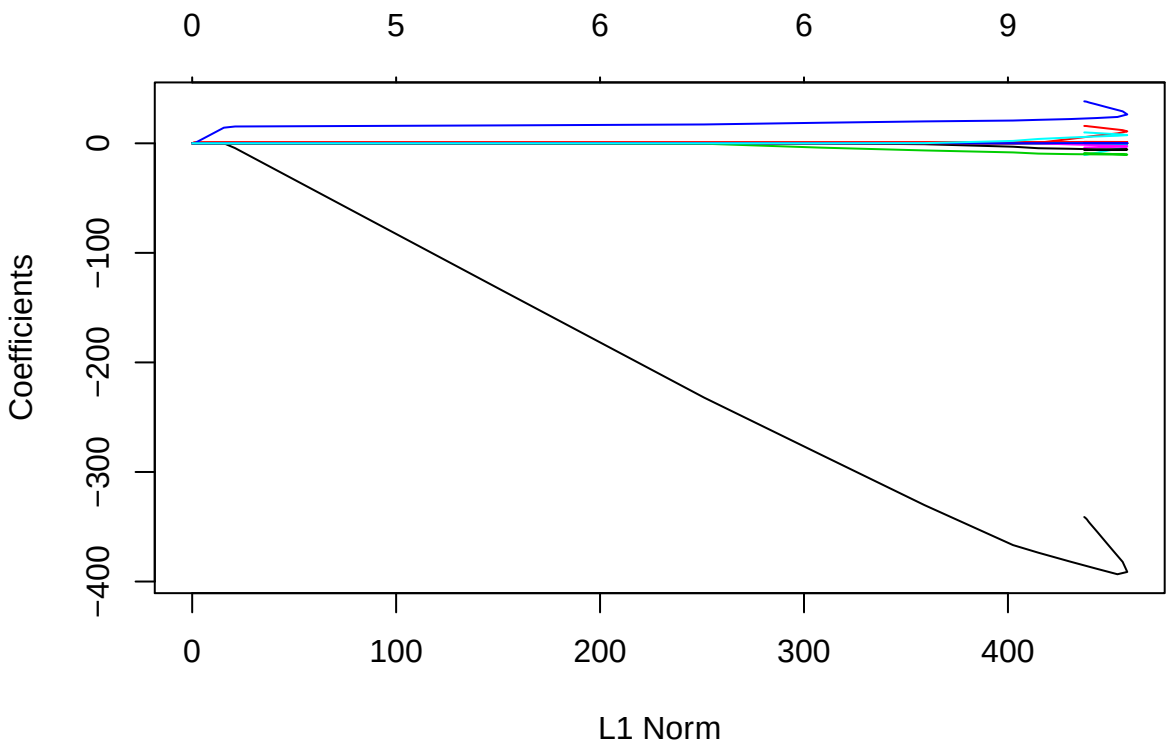

```
bestlam = cv.out$lambda.min
ridge_pred <- predict(ridge_mod, s = bestlam, newx = x_test)
ridge_mse<-mean((ridge_pred - y_test)^2)
ridge_mse
```

```
## [1] 1665026
```

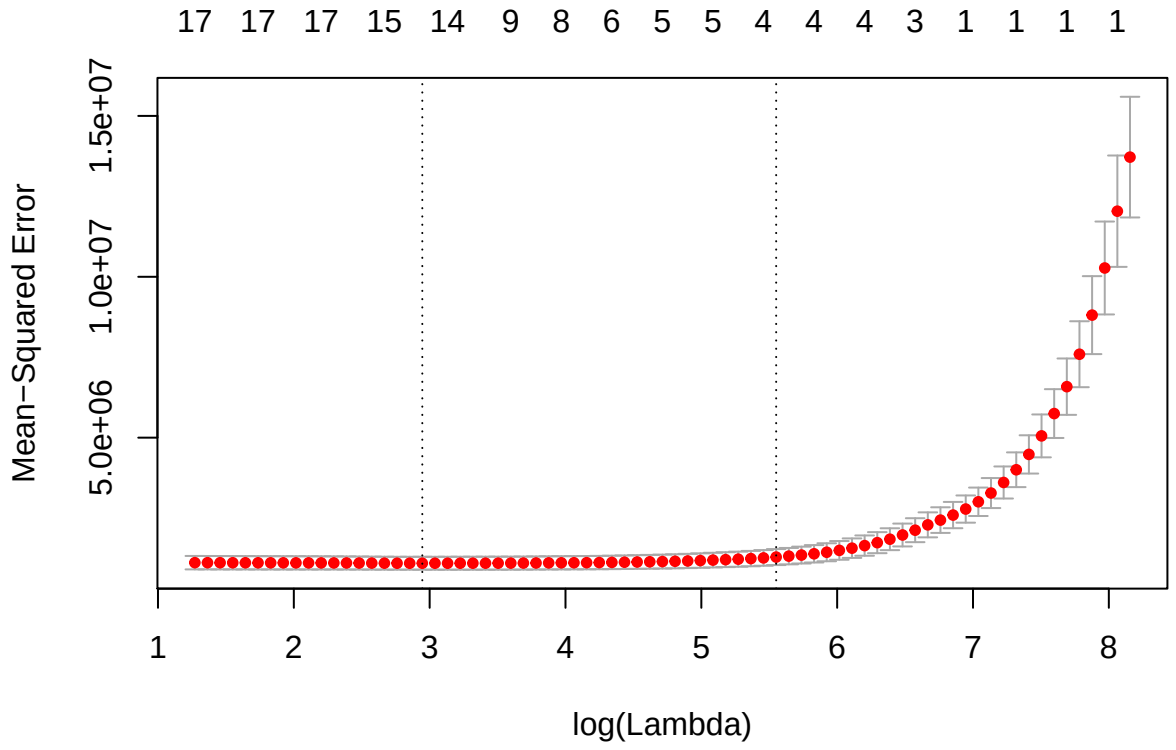
The Test MSE is 1665026. This is higher than the test set MSE of the null model and of least squares.

D

```
lasso_mod <- glmnet(x_train, y_train,
alpha = 1,
lambda = grid) # Fit lasso model on training data
plot(lasso_mod)
```




```
set.seed(11)
cv.out = cv.glmnet(x_train, y_train, alpha = 1)
plot(cv.out)
```




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

```r
lasso_pred <- predict(lasso_mod, s = bestlam, newx = x_test)

lasso_mse<-mean((lasso_pred - y_test)^2)
lasso_mse
```

```
## [1] 1635158
```

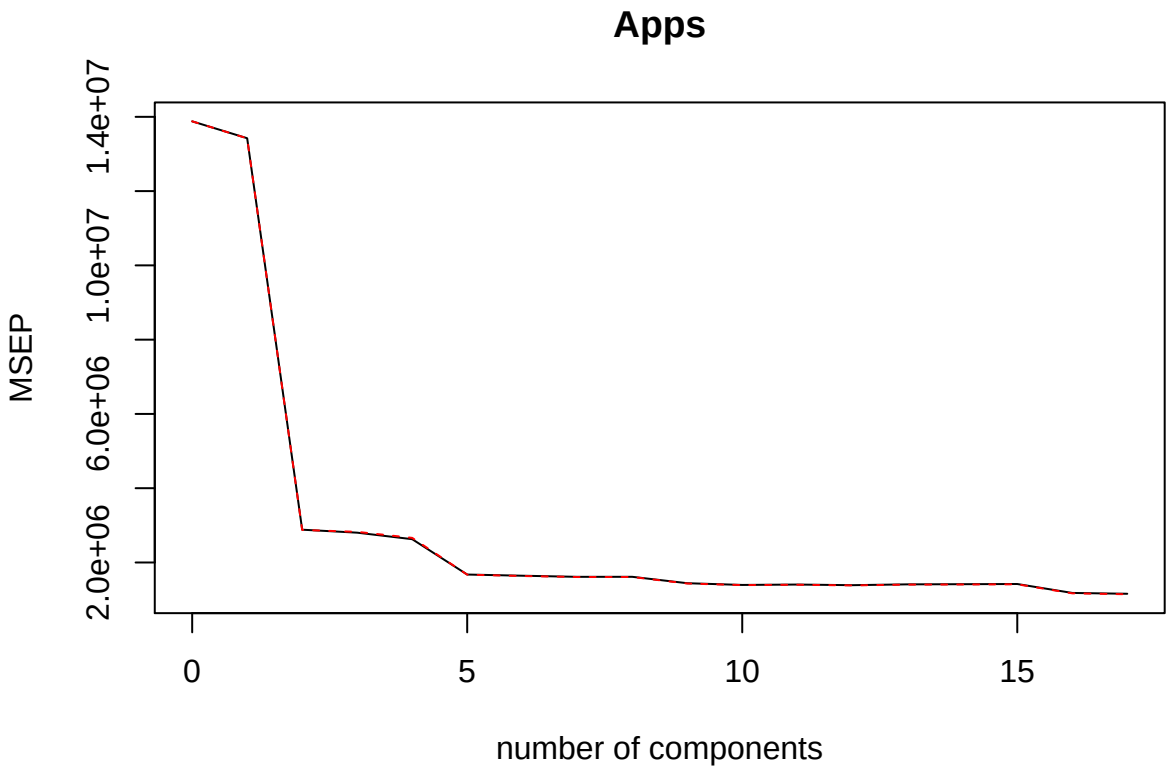
```r
out = glmnet(x, y, alpha = 1, lambda = grid) # Fit lasso model on full dataset
# Display coefficients using lambda chosen by CV
lasso_coef = predict(out, type = "coefficients", s = bestlam)
lasso_coef
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -1.001915e+03
## (Intercept)  .
## Accept       1.475960e+00
## Enroll      -1.958140e-01
## Top10perc    3.529903e+01
## Top25perc   -3.726115e+00
## F.Undergrad  .
## P.Undergrad  3.366729e-02
## Outstate    -8.313713e-02
## Room.Board   1.138918e-01
## Books        .
## Personal     3.436299e-03
## PhD         -4.221560e+00
## Terminal    -1.454712e+00
## S.F.Ratio    1.288507e+01
## perc.alumni -2.138175e+00
## Expend       7.417485e-02
## Grad.Rate    4.938139e+00
```

The Test MSE is 1635158. This is higher than the test set MSE of the null model and of least squares and but lower than ridge regression. There are 15 non-zero coefficient estimates.

E

```r
pcr_mod<- pcr(Apps~., data = train_College, scale = TRUE, validation = "CV")
validationplot(pcr_mod, val.type = "MSEP")
```



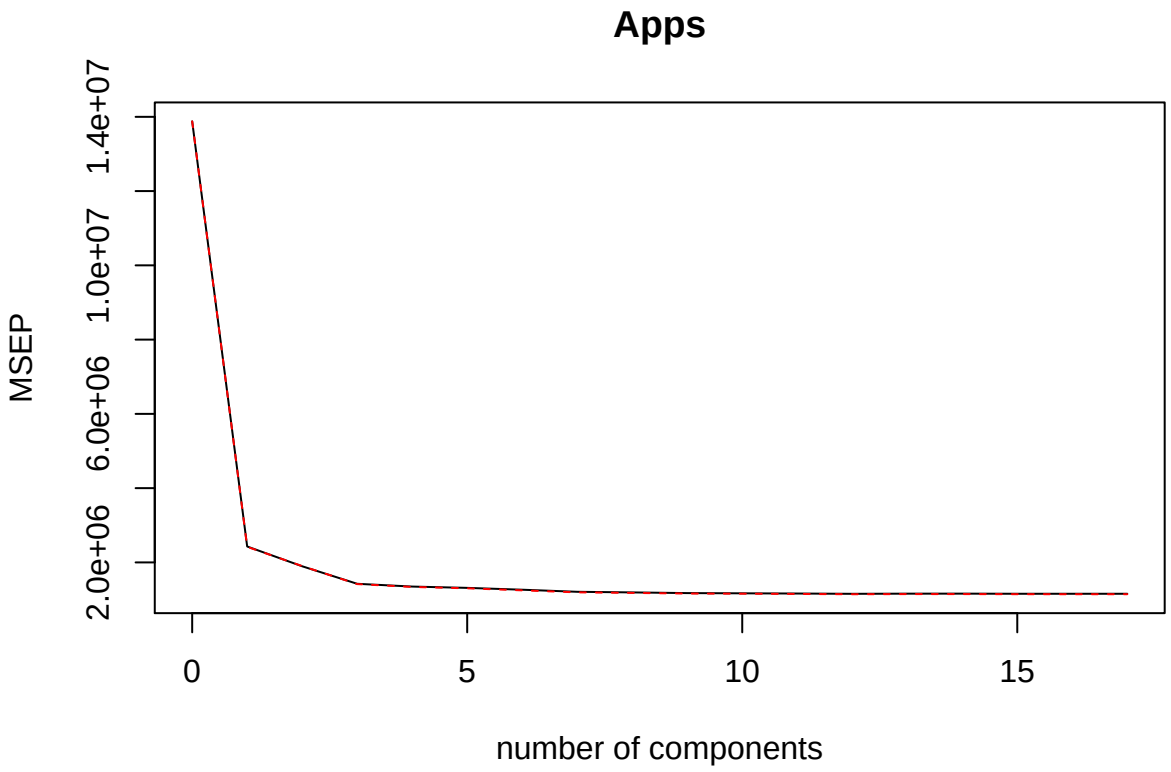

```
pcr_pred <- predict(pcr_mod, x_test, ncomp= 17)
pcr_mse <- mean((pcr_pred-y_test)^2)
pcr_mse
```

```
## [1] 1538442
```

We find that the lowest cross-validation error occurs when M = 17 components are used. We compute the test MSE to be 1538442

F

```
set.seed(11)
pls = plsr(Apps~., data = train_College, scale = TRUE, validation = "CV")
validationplot(pls, val.type = "MSEP")
```




```
set.seed(11)
pls_pred <- predict(pls, x_test, ncomp = 10)
pls_mse<-mean((pls_pred - y_test)^2)
pls_mse
```

```
## [1] 1508987
```

We find that the lowest cross-validation error occurs when M = 10 components are used. We computed that the test MSE is 1508987

G

PLS seems to have the best test error rate and compared to other methods and PCR seems to have a similar MSE. Least-squares, ridge regression and Lasso also seem have very similar error rates.

```
avg <- mean(test_College$Apps)
lS_R <- 1 -  lm_mse/ mean((avg - test_College$Apps)^2)
lS_R
```

```
## [1] 0.9044281
```

```
ridge_R <- 1 - ridge_mse / mean((avg - test_College$Apps)^2)
ridge_R
```

```
## [1] 0.8965644
```

```
lasso_R<- 1 -  lasso_mse/ mean((avg - test_College$Apps)^2)
lasso_R
```

```
## [1] 0.8984199
```

```r
pcr_R <- 1 - pcr_mse/ mean((avg - test_College$Apps)^2)
pcr_R
```

```
## [1] 0.9044281
```

```r
pls_R <- 1 - pls_mse / mean((avg - test_College$Apps)^2)
pls_R
```

```
## [1] 0.9062579
```

All methods have a 89.6%-90% accuracy rate.