Q3 part a)

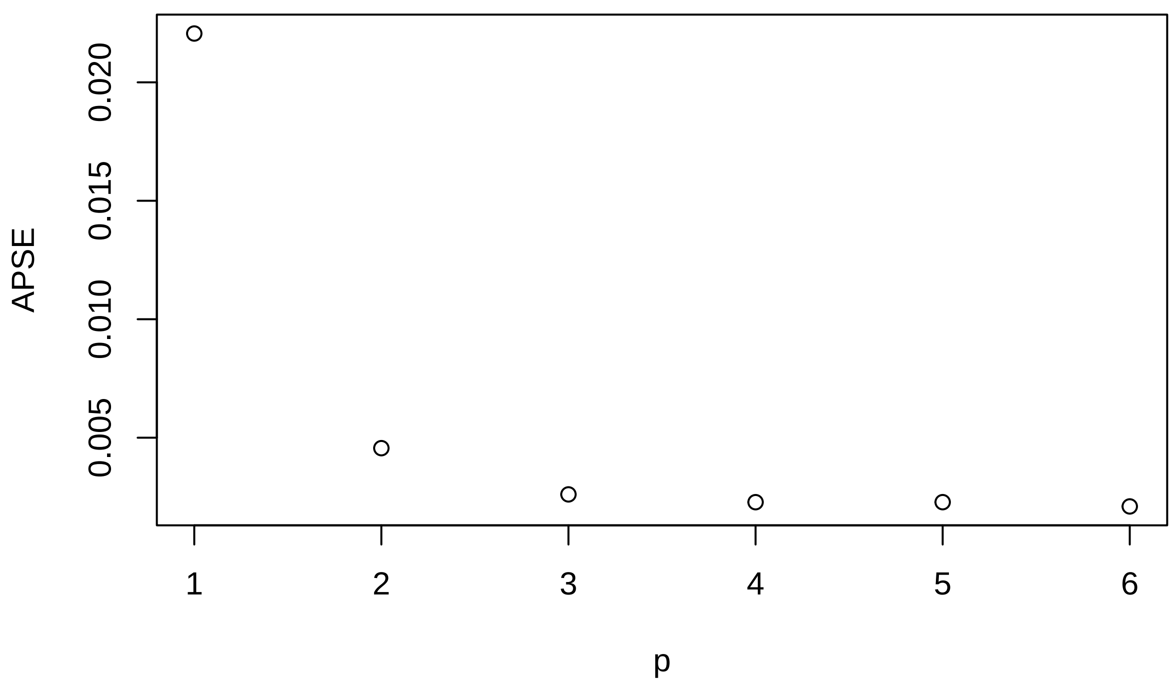
```
data <- read.csv("Quality.csv")
Y <- ts(data)
t <- as.vector(time(Y))

t.test = c(5, 14, 29, 30, 36, 39, 66, 71, 79, 84, 85, 96, 109,
              112, 135, 136, 139, 146, 156, 171)
t.train = t[-t.test]
y.train = Y[t.train]
y.test = Y[t.test]

poly.Time = poly(t, 6)
modelMatrix <- model.matrix(~ poly.Time) # 180x15

APSE = c()
for(p in 1:6){
  X = modelMatrix[t.train, 1:p+1]
  model <- lm(y.train~X)
  Xpred = modelMatrix[t.test, 1:p+1]
  Xpred <- cbind(1, Xpred)
  coffs <- c(model$coefficients)
  pred = Xpred%*%coffs
  APSE[p] = mean((pred-y.test)^2)
}

p=c(1:6)
plot(x=p, y=APSE)
```




```
which.min(APSE)
```

```
## [1] 6
```

Based on APSE, the optimal degree polynomial which minimizes APSE is $p = 6$. I think the simpler model for $p = 4$ will have similar performance since it is similar to the APSE value for $n = 6$ and it corresponds to a local minimum ($APSE(5) < APSE(4) < APSE(3)$).

Question 3 part b)

```
library(glmnet())
```

```
## Loading required package: Matrix
## Loaded glmnet 4.1-8
```

```
Y <- ts(data)
t <- as.vector(time(Y))
poly.Time = poly(t, 15)

X <- model.matrix(~ poly.Time) # 180x15
y <- as.vector(Y) # 180x1


Time.test = c(5, 14, 29, 30, 36, 39, 66, 71, 79, 84, 85, 96, 109,
              112, 135, 136, 139, 146, 156, 171)
Time.train = t[-Time.test]

X.train <- X[Time.train, 2:16]
X.test <- X[Time.test, 2:16]

y.train <-y[Time.train]
y.test <- y[Time.test]

Log.Lambda.Seq = seq(-7, 3, by = 0.1)
Lambda.Seq = c(0, exp(Log.Lambda.Seq))



### ELASTIC NET models
## alpha = 0, p=9
X.train.alpha0 <- X.train[, 1:9]
model.alpha0 = cv.glmnet(X.train.alpha0, y.train, type.measure="mse", lambda= Lambda.Seq, alpha=0, famil
alpha0.coeffs = matrix(predict(model.alpha0,type="coef"),ncol=1)

X.test.alpha0 <- X.test[, 1:9]
prediction.alpha0 = cbind(1,X.test.alpha0)%*%alpha0.coeffs
APSE_alpha0 = mean((prediction.alpha0-y.test)^2)


## alpha = 0.5, p=7
X.train.alpha05 <- X.train[, 1:7]
model.alpha05 = cv.glmnet(X.train.alpha05, y.train, type.measure="mse", lambda= Lambda.Seq, alpha=0.5, 
alpha05.coeffs = matrix(predict(model.alpha05,type="coef"),ncol=1)

X.test.alpha05 <- X.test[, 1:7]
prediction.alpha05 = cbind(1,X.test.alpha05)%*%alpha05.coeffs
APSE_alpha05 = mean((prediction.alpha05-y.test)^2)

## alpha = 1, p=8
X.train.alpha1 <- X.train[, 1:8]
model.alpha1 = cv.glmnet(X.train.alpha1, y.train, type.measure="mse", lambda= Lambda.Seq, alpha=1, famil
alpha1.coeffs = matrix(predict(model.alpha1,type="coef"),ncol=1)
```

```r
X.test.alpha1 <- X.test[, 1:8]
prediction.alpha1 = cbind(1,X.test.alpha1)%*%alpha1.coeffs
APSE_alpha1 = mean((prediction.alpha1-y.test)^2)


"OLS REGRESSION MODEL:"
```

```
## [1] "OLS REGRESSION MODEL:"
```

```r
APSE[6]
```

```
## [1] 0.002099335
```

```r
"REGULARIZED REGRESSION MODELS"
```

```
## [1] "REGULARIZED REGRESSION MODELS"
```

```r
APSE_alpha0
```

```
## [1] 0.00210101
```

```r
APSE_alpha05
```

```
## [1] 0.002127338
```

```r
APSE_alpha1
```

```
## [1] 0.002074784
```

Based on APSE, the best predictive model is elastic net $\alpha = 0.5$

Question 3 part c)

The comparison is not fair. In elastic net models, p was more carefully chosen with cross-validation.