Homework 2

Brett Burk

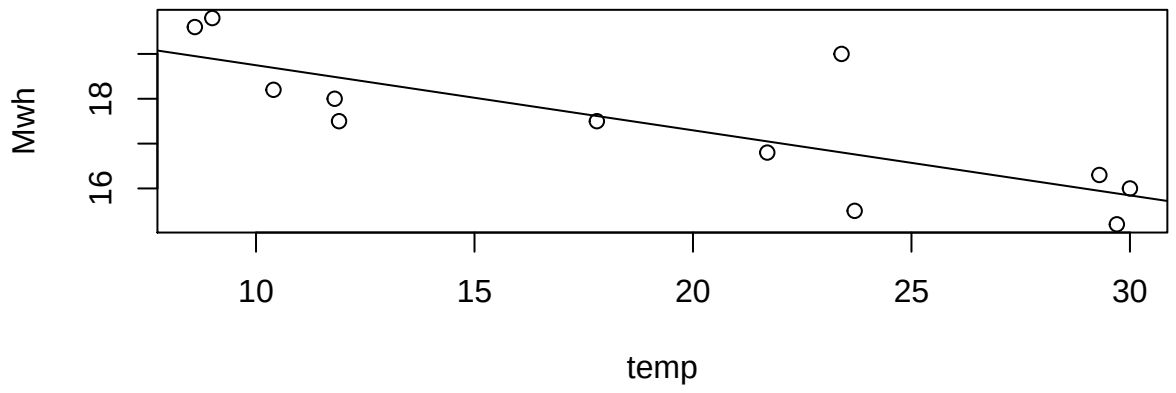
Saturday, June 16, 2015

```
library(fma)
library(ggplot2)
library(reshape2)
library(caret)
library(AppliedPredictiveModeling)
library(pls)
library(elasticnet)
```

Hyndman and Athanasopoulos

Question 4.1a objective: determine whether there is and why there might be a negative relationship

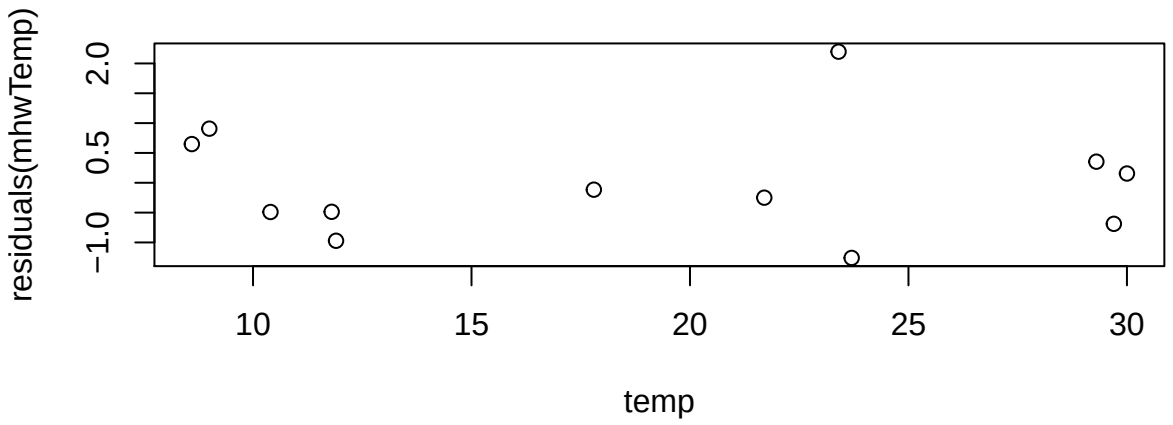
```
plot(Mwh ~ temp, data = econsumption)
mhwTemp <- lm(Mwh ~ temp, data = econsumption)
abline(mhwTemp)
```



One would assume there's a negative relationship because at colder temperatures people would be more likely to be inside using electricity and/or heating their homes.

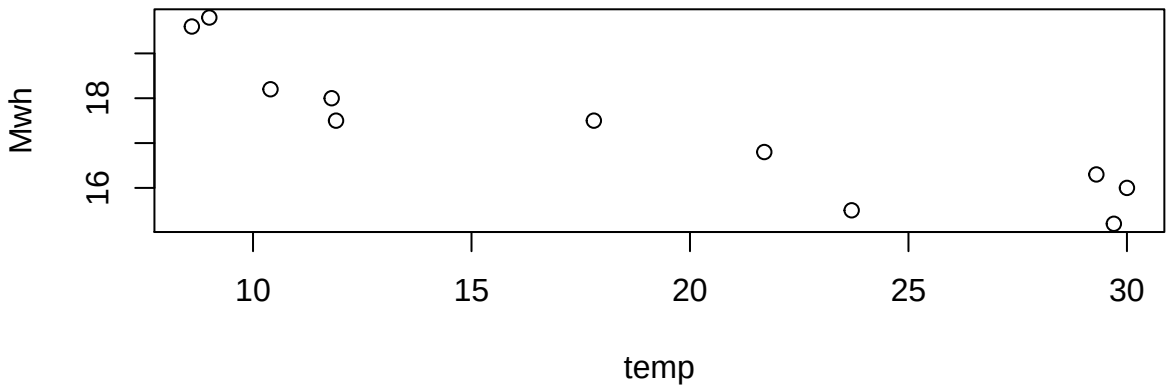
Question 4.1b objective: determine if there are influential observations/outliers

```
plot(residuals(mhwTemp) ~ temp, data = econsumption)
```

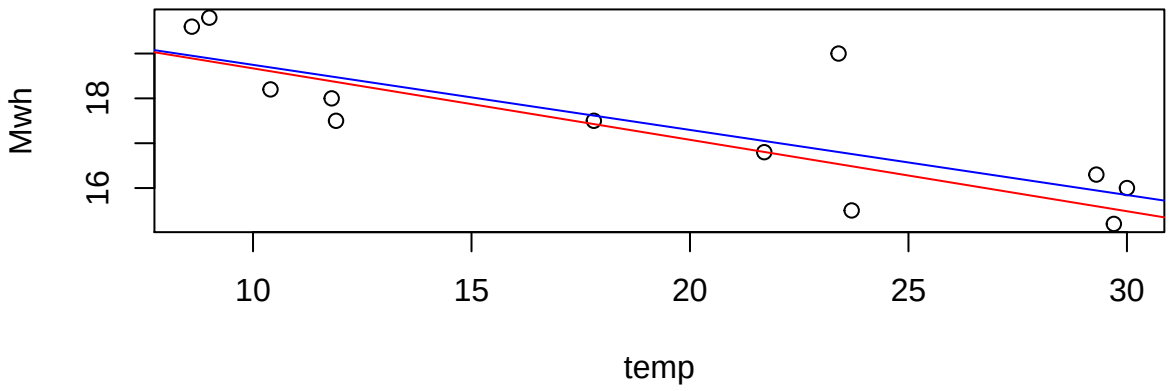


The model seems relatively fair, although there is one strong outlier–(19.0, 23.4) that affects the model. We can also see that it most correctly predicts middle temperatures. It looks better after the single point is removed:

```
econsumptionClean <- econsumption[-8,]
plot(Mwh ~ temp, data = econsumptionClean)
```




```
fitClean <- lm(Mwh ~ temp, data = econsumptionClean)
plot(Mwh ~ temp, data = econsumption)
abline(mhwTemp, col = 'Blue')
abline(fitClean, col = 'Red')
```



There is a change, but it's not drastic.

Question 4.1c objective: make predictions and check their feasability

```
forecast(mhwTemp, 10)$mean
```

```
## [1] 18.74795
```

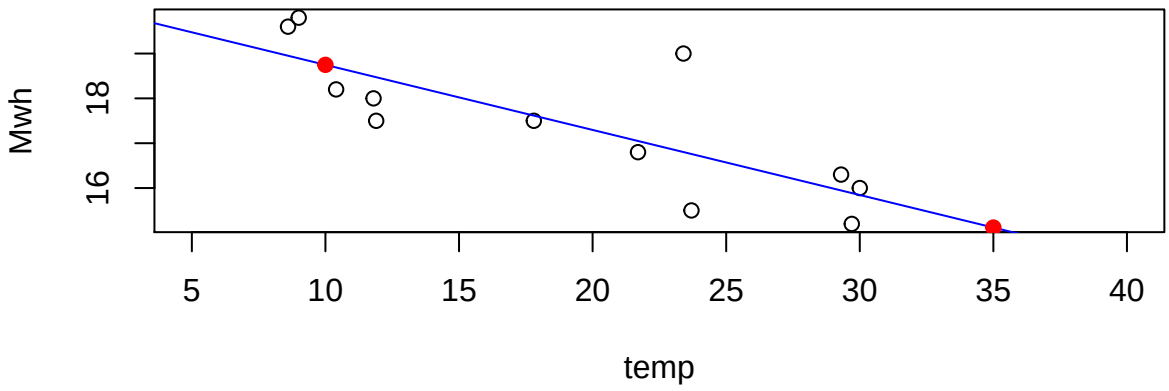
```
forecast(mhwTemp, 35)$mean
```

```
## [1] 15.11902
```

```
c(forecast(mhwTemp, 10)$mean, forecast(mhwTemp, 35)$mean)
```

```
## [1] 18.74795 15.11902
```

```
plot(Mwh ~ temp, data = econsumption, xlim = c(5, 40))
abline(mhwTemp, col = 'Blue')
points(c(10, 35), c(forecast(mhwTemp, 10)$mean, forecast(mhwTemp, 35)$mean), pch = 19, col = 'Red')
```



The first prediction (that for 10) seems pretty reasonable, and while the second is not outlandish, the fact that we are no longer making predictions within observed variables is obviously problematic, as we should not extrapolate our prediction to points outside of the observed interval.

Question 4.1d objective: Identify the prediction intervals
95% interval for 10°C

```
c(forecast(mhwTemp, 10)$lower[2], forecast(mhwTemp, 10)$upper[2])
```

```
## [1] 16.34824 21.14766
```

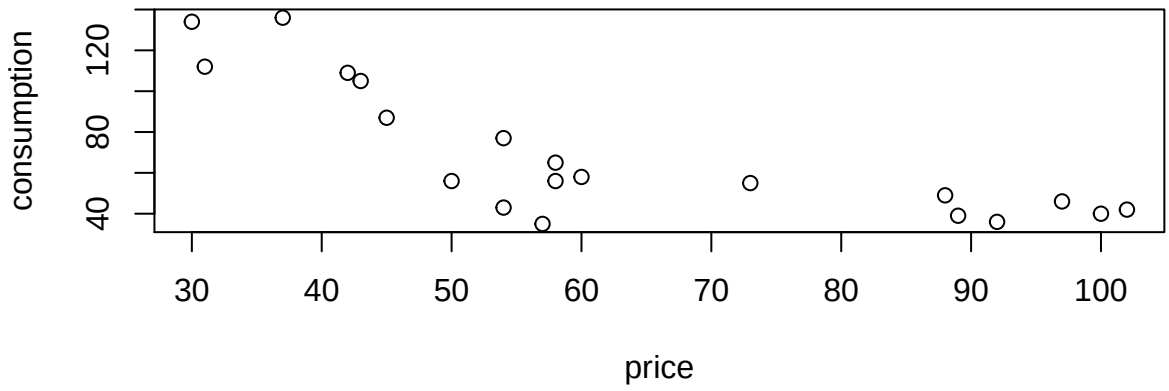
95% interval for 35°C

```
c(forecast(mhwTemp, 35)$lower[2], forecast(mhwTemp, 35)$upper[2])
```

```
## [1] 12.49768 17.74035
```

Question 5.2a objective: Plot the data to get an idea of what it looks like.

```
plot(consumption ~ price, data = texasgas)
```

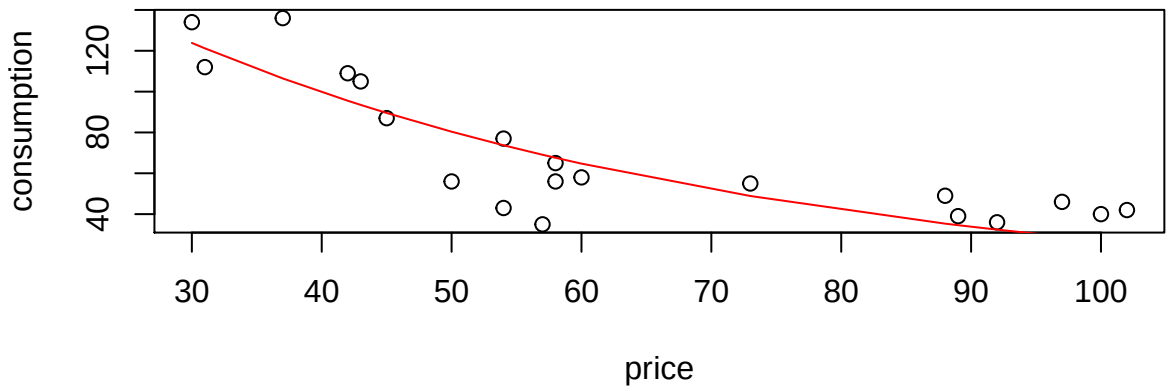


Question 5.2b The line changes with the price because consumption is likely to go down with higher prices, some things require electricity no matter what (and some people may be more expensive to serve, such as farmers, vs. residential consumption).

Question 5.2c objective: Plot the three models, and find the coefficients and residual variance.

Exponential:

```
plot(consumption ~ price, data = texasgas)
expFit <- nls(consumption ~ exp(a + b * price),
              data = texasgas,
              start = list(a = 0, b = 0))
lines(texasgas$price, predict(expFit, x = texasgas$price), col = 'red')
```



Coefficients followed by the variance of residuals:

```
summary(expFit)$coefficients
```

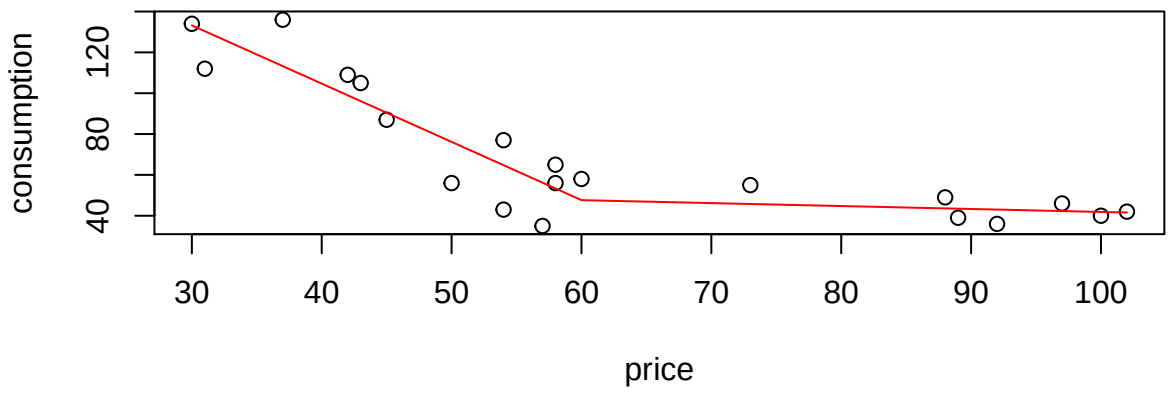
```
##      Estimate  Std. Error   t value     Pr(>|t|)
## a  5.46772919 0.160962468 33.968970 8.887222e-18
## b -0.02162313 0.003295812 -6.560789 3.648543e-06
```

```
var(resid(expFit))
```

```
## [1] 274.4557
```

Piecewise:

```
pricep <- pmax(texasgas$price - 60, 0)
pieceFit <- lm(consumption ~ price + pricep, data = texasgas)
x <- min(texasgas$price):max(texasgas$price)
z <- pmax(x-60, 0)
fcast <- forecast(pieceFit, newdata = data.frame(price = x, pricep = z))
plot(consumption ~ price, data = texasgas)
lines(x, fcast$mean, col = 'red')
```



Coefficients followed by the variance of residuals:

```
summary(pieceFit)$coefficients
```

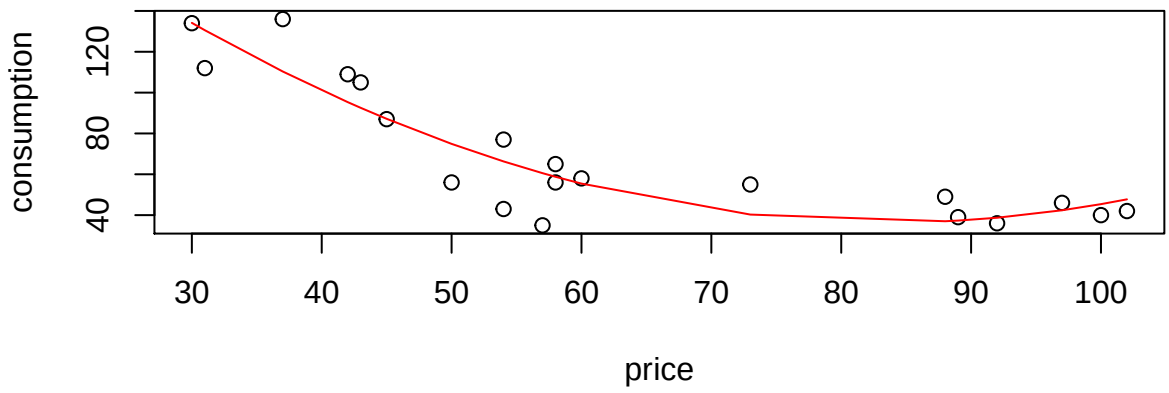
```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 218.826325 17.4986186 12.505349 5.339882e-10
## price        -2.853377  0.3560114 -8.014848 3.558902e-07
## pricep        2.709179  0.5144457  5.266210 6.302422e-05
```

```
var(resid(pieceFit))
```

```
## [1] 156.6595
```

Polynomial:

```
polyFit <- nls(consumption ~ a + b * price + c * price^2,
               data = texasgas,
               start = list(a = 0, b = 0, c = 0))
plot(consumption ~ price, data = texasgas)
lines(texasgas$price, predict(polyFit, x = texasgas$price), col = 'red')
```



Coefficients followed by the variance of residuals:

```
summary(polyFit)$coefficients
```

```
##       Estimate Std. Error   t value     Pr(>|t|)
## a 273.93062841 31.0316141  8.827470 9.320316e-08
## b  -5.67586264  1.0090863 -5.624754 3.031520e-05
## c   0.03390386  0.0074119  4.574246 2.694526e-04
```

```
var(resid(polyFit))
```

```
## [1] 184.7879
```

Question 5.2d Exponential:
First the R^2

```
1-sum(residuals(expFit)^2)/(length(texasgas$consumption) * var(texasgas$consumption))
```

```
## [1] 0.7613988
```

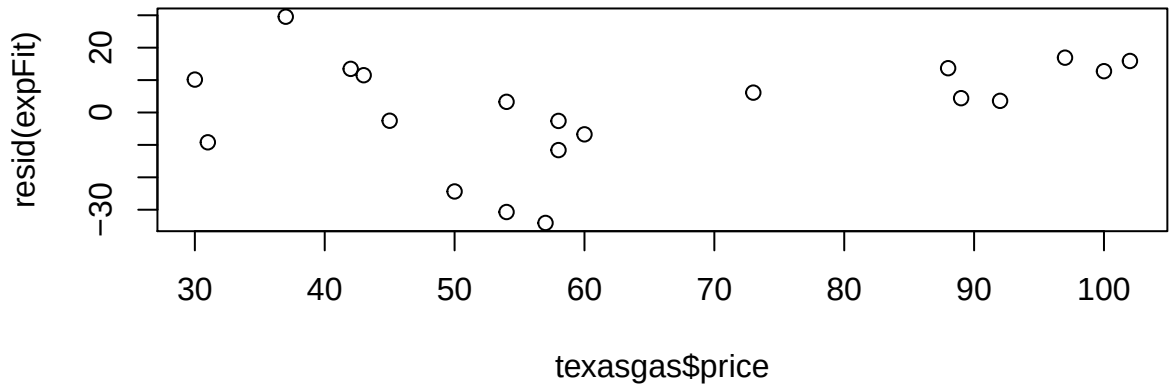
AIC

```
AIC(expFit)
```

```
## [1] 174.1002
```

Residuals plot:

```
plot(texasgas$price, resid(expFit))
```



This model is far from perfect, it fits a lot of the data points pretty well, but there are a few points that it drastically over/underestimates. It works well on the later data, but is not good at the data before 60 degrees.

Piecewise:
R^2

```
1-sum(residuals(pieceFit)^2)/(length(texasgas$consumption) * var(texasgas$consumption))
```

```
## [1] 0.8643006
```

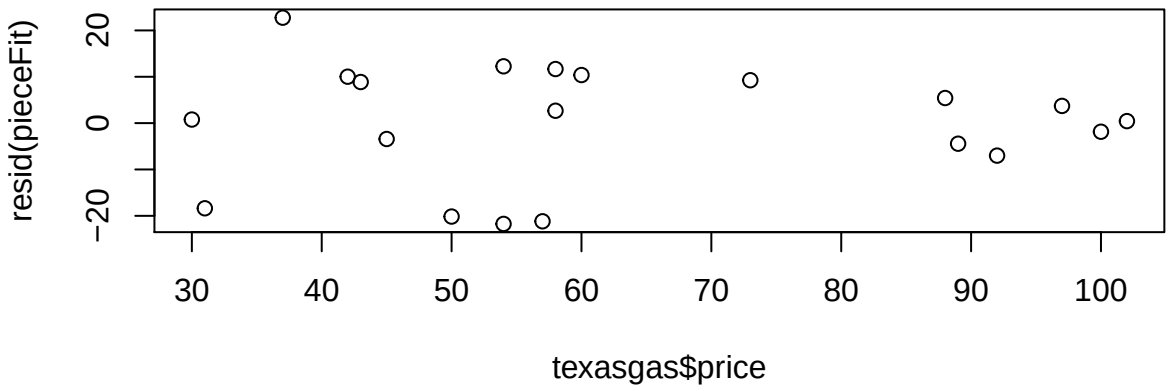
AIC

```
AIC(pieceFit)
```

```
## [1] 164.8132
```

Residuals plot:

```
plot(texasgas$price, resid(pieceFit))
```



This is the best model of the three, again, it is not perfect, but most of the data points float around it. It doesn't do a great job on the first half of the data, but it doesn't do terribly either.

Polynomial:
R^2

```
1-sum(residuals(polyFit)^2)/(length(texasgas$consumption) * var(texasgas$consumption))
```

```
## [1] 0.8399356
```

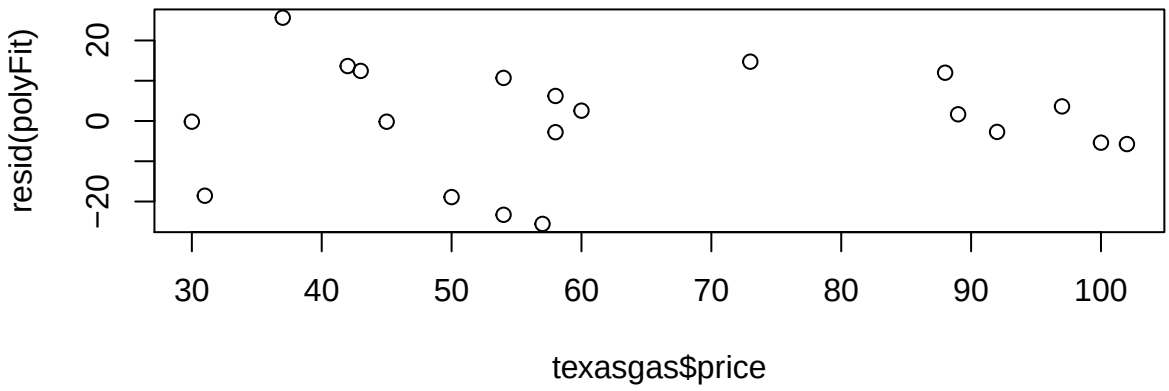
AIC

```
AIC(polyFit)
```

```
## [1] 168.1158
```

Residuals plot:

```
plot(texasgas$price, resid(polyFit))
```



This model is middle of the road, it works, and it does a better job with the first half of the data, although it doesn't do as well of a job estimating the second half.

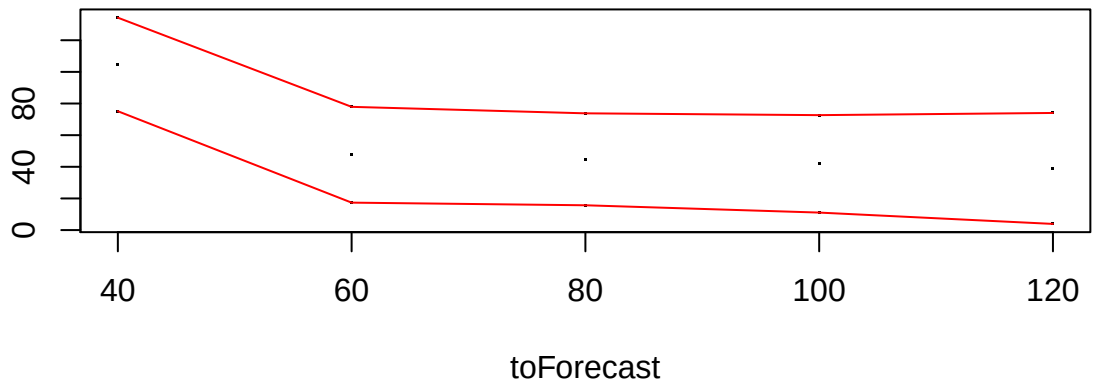
Question 5.2e objective: Use the best model from above to make estimates

```
toForecast <-seq(40, 120, 20)
pricep2 <- pmax(toForecast - 60, 0)
fcast <- forecast(pieceFit, newdata = data.frame(price = toForecast, pricep = pricep2))
cbind(toForecast, fcast$mean)
```

```
##   toForecast
## 1         40 104.69124
## 2         60  47.62370
## 3         80  44.73974
## 4        100  41.85579
## 5        120  38.97183
```

Question 5.2f objective: Determine and graph the prediction intervals

```
foreMeans <- cbind(toForecast, fcast$mean)
foreUpper <- cbind(toForecast, fcast$upper[,2])
foreLower <- cbind(toForecast, fcast$lower[,2])
foreConf <- rbind(foreMeans, foreUpper, foreLower)
plot(foreConf, pch = '.')
lines(foreUpper, col = 'red')
lines(foreLower, col = 'red')
```



We can be 95% confident that our predictions for those values (the extrapolation of using lines rather than points should not be interpreted to relate to lines, but rather to upper and lower limits of the points therein) will lie between the red lines. This means that we would expect about one out of every twenty such obeservations to lie outside of those lines.

Question 5.2g objective: Show the relationship between P^2 and P.
P^2 is obviously completely dependent on P, which mmay lead it to have the issues that come along with variables that are highly dependent on each other: wide confidence intervals, and large standard errors.

Kuhn and Johnson

Question 6.2a objective: Load in the data.

```
data(permeability)
```

Question 6.2b objective: Eliminate unnecessary predictors and give the amount of predictors afterwards.

```
fingerZero <- fingerprints[,nearZeroVar(fingerprints)]
ncol(fingerZero)
```

```
## [1] 719
```

Question 6.2c objective: Split the data and determine the ideal number of components and the R^2

```
set.seed(06202015)
# Sampling
chosen <- sample(1:nrow(fingerprints))

# Creating the training set
chosenTrain <- chosen[1:(length(chosen) * 0.8)]
fingerTrain <- fingerprints[chosenTrain,]
```

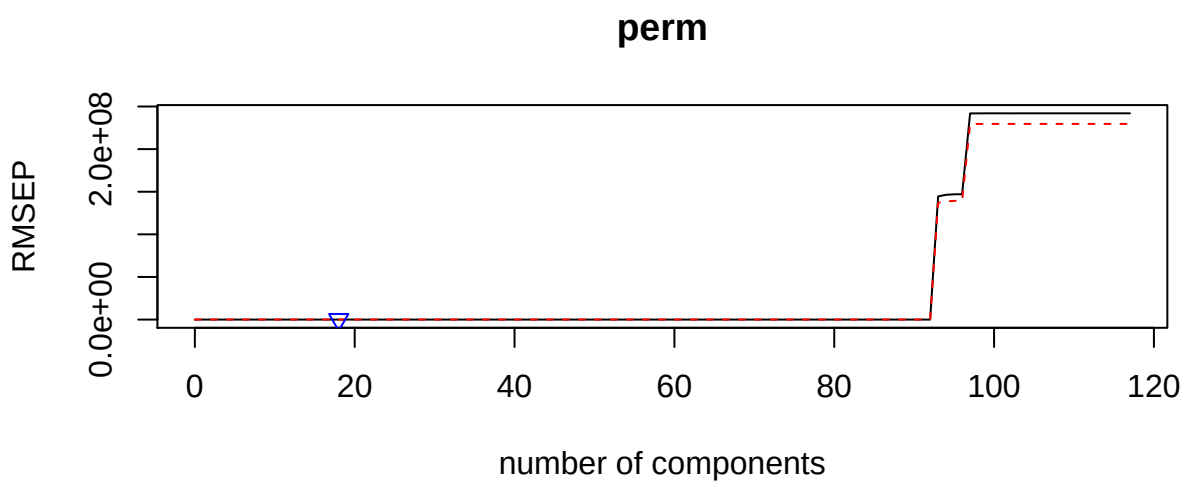
```
# Preprocessing
nzv <- nearZeroVar(fingerTrain)
fingerTrain <- fingerTrain[,-nzv]
fingerTrain <- data.frame(fingerTrain, perm = permeability[chosenTrain])

# Creating the test set
chosenTest <- chosen[(length(chosen) * 0.8 + 1):length(chosen)]
fingerTest <- fingerprints[chosenTest,]
# Using the same colums as the training set
fingerTest <- fingerTest[,-nzv]
fingerTest <- data.frame(fingerTest, perm = permeability[chosenTest])

# Training the model
plsFit <- plsr(perm ~ ., data = fingerTrain, validation = 'CV')
```

The amount of ideal components:

```
preds <- predict(plsFit, fingerTrain)
mincomp <- which.min(RMSEP(plsFit)$val)
validationplot(plsFit, val.type = 'RMSEP')
points(mincomp, RMSEP(plsFit)$val[mincomp], col = 'Blue', pch = 25)
```



22 Seems like the correct number.

```
finalPred <- predict(plsFit, fingerTrain, ncomp = 22)
plsEval <- data.frame(obs = fingerTrain$perm, pred = finalPred)

# For some reason this broke, so this is a hacky fix
names(plsEval)[2] <- 'pred'

defaultSummary(plsEval)
```

```
##      RMSE  Rsquared
## 4.7760020 0.9110585
```

Question 6.2d objective: Apply the model to the test data set and evaluate the R^2

```
testPred <- predict(plsFit, fingerTest, ncomp = 22)
plsTestEval <- data.frame(obs = fingerTest$perm, pred = testPred)
names(plsTestEval)[2] <- 'pred'
defaultSummary(plsTestEval)
```

```
##      RMSE  Rsquared
## 11.479116  0.398136
```

Because of the low R^2 here compared to the training set, we may have overfit our model in the first part.

Question 6.2e objective: Try other models and test them First I tried ordinary least squares regression (still having used PCA)

```
olrFit <- lm(perm ~ ., data = fingerTrain)
summary(olrFit)$r.squared
```

```
## [1] 0.9632673
```

```
defaultSummary(data.frame(obs = fingerTest$perm, pred = predict(olrFit, fingerTest)))['Rsquared']
```

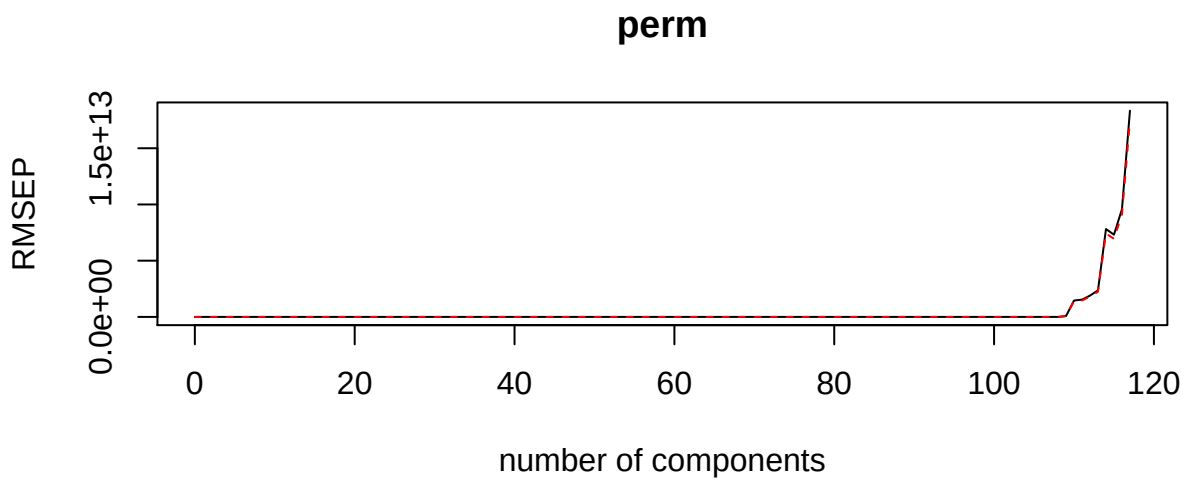
```
## Warning in predict.lm(olrFit, fingerTest): prediction from a rank-deficient
## fit may be misleading
```

```
##  Rsquared
## 0.1490602
```

The predicted performance is much worse for an ordinary least squares than a partial least squares. The error here is also pertinent, as this data set is much wider than it is long.

Next I tried PCR

```
pcrTrain <- data.frame(fingerprints[chosenTrain,], perm = permeability[chosenTrain])
pcrTest <- data.frame(fingerprints[-chosenTrain,], perm = permeability[-chosenTrain])
pcrModel <- pcr(perm ~ ., data = pcrTrain, validation = 'CV')
validationplot(pcrModel, val.type = 'RMSEP')
```




```
which.min(RMSEP(pcrModel)$val)
```

```
## [1] 62
```

```
pcrModel <- pcr(perm ~ ., data = pcrTrain, validation = 'CV', ncomp = 62)
pcrComparison <- data.frame(obs = pcrTrain$perm, pred = predict(pcrModel, pcrTrain, ncomp = 62))
names(pcrComparison)[2] <- 'pred'
defaultSummary(pcrComparison)['Rsquared']
```

```
##  Rsquared
## 0.8177825
```

```
pcrTestComparison <- data.frame(obs = pcrTest$perm, pred = predict(pcrModel, pcrTest, ncomp = 62))
names(pcrTestComparison)[2] <- 'pred'
defaultSummary(pcrTestComparison)['Rsquared']
```

```
##  Rsquared
## 0.4472025
```

Which definitely had better results.

Question 6.2f I would definitely not recommend OLS over PLS, however, PCR definitely seems like a better option than PLS. The R^2 is better for this model, although a more complex model is needed in order to achieve better results on the test set.