# Week 3 Homework

*Ben Arancibia*

*June 23, 2015*

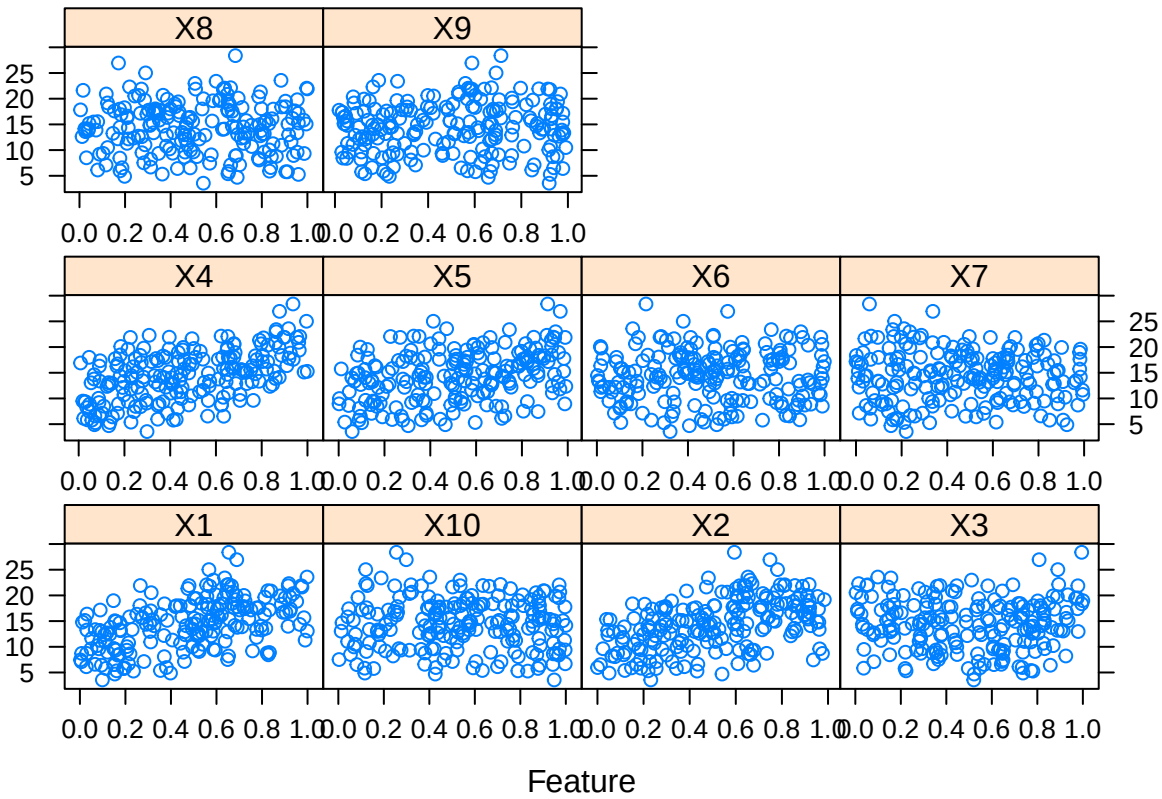
**KJ 7.2** Friedman (1991) introduced several benchmark datasets create by simulation. One of these simulations used the following non–linear equation to create data:

$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + N(0, \sigma^2)$

```
library(mlbench)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.1.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
set.seed(200)
trainingData <- mlbench.friedman1(200, sd = 1)
trainingData$x <- data.frame(trainingData$x)
featurePlot(trainingData$x, trainingData$y)
```



```
testData <- mlbench.friedman1(5000, sd = 1)
testData$x <- data.frame(testData$x)
```

Tune several models on these data. For example:

```
set.seed(921)
knnModel <- train(x = trainingData$x, y = trainingData$y, method = "knn",
                  preProc = c("center", "scale"),
                  tuneLength = 10)
knnModel
```

```
## k-Nearest Neighbors
##
## 200 samples
##  10 predictor
##
## Pre-processing: centered, scaled
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...
##
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   RMSE SD    Rsquared SD
##    5  3.488933  0.5019753  0.2658769  0.07412999
##    7  3.324957  0.5484600  0.2275136  0.06432697
##    9  3.224541  0.5853589  0.2425946  0.06903863
##   11  3.178450  0.6091595  0.2593909  0.07304742
##   13  3.183655  0.6183553  0.2523970  0.06995701
##   15  3.188007  0.6250729  0.2408867  0.06604822
##   17  3.214343  0.6281943  0.1890541  0.05315353
##   19  3.208743  0.6403733  0.1921773  0.05056336
##   21  3.215199  0.6487431  0.1830866  0.04961129
##   23  3.235167  0.6528070  0.1870124  0.04751686
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was k = 11.
```

```
knnPred <- predict(knnModel, newdata = testData$x)
postResample(pred = knnPred, obs = testData$y)
```

```
##      RMSE  Rsquared
## 3.1222641 0.6690472
```

Which models appear to give the best performance? Does MARS select the informative predictors (those named X1–X5)?

K-nearest neighbors models are better when predictors and the response relies on the samples' proximity in the predictor space. This data does not have geographic information so another model might perform better. Two models to look at are MARS and SVM.

**MARS Model**

```
marsGrid <- expand.grid(degree = 1:2, nprune = seq(2,14,by=2))
set.seed(921)
marsModel <- train(x = trainingData$x, y = trainingData$y, method = "earth",
                   preProc = c("center", "scale"),
                   tuneGrid = marsGrid)
```

```
## Loading required package: earth
```

```
## Warning: package 'earth' was built under R version 3.1.3
```

```
## Loading required package: plotmo
```

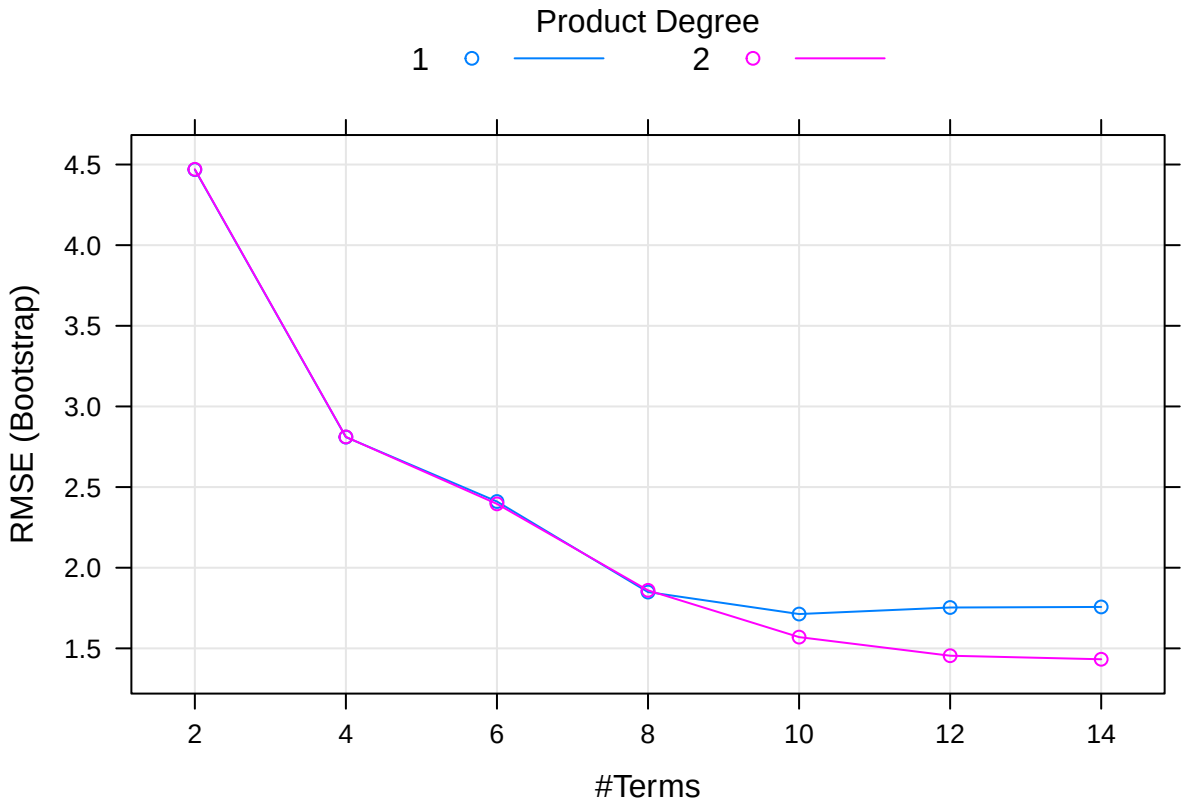
```
## Warning: package 'plotmo' was built under R version 3.1.3
```

```
## Loading required package: plotrix
```

```
## Warning: package 'plotrix' was built under R version 3.1.3
```

```
## Loading required package: TeachingDemos
```

```
marsPred <- predict(marsModel, newdata = testData$x)

plot(marsModel)
```



```
postResample(pred = marsPred, obs = testData$y)
```

```
##      RMSE  Rsquared
## 1.2779993 0.9338365
```

```
marsModel$bestTune$nprune
```

```
## [1] 14
```

```
marsModel$bestTune$degree
```

```
## [1] 2
```

```
round(getTrainPerf(marsModel)[1, "TrainRMSE"], 2)
```

```
## [1] 1.43
```

The above plot shows the MARS tuning. There are 14 terms, with degree of 2 and RMSE of 1.43%. To determine which predictors are used in the final model, call the variable importance scores.

```
varImp(marsModel)
```

```
## earth variable importance
##
##     Overall
## X1   100.00
## X4    84.98
## X2    68.87
## X5    48.55
## X3    38.96
## X9     0.00
## X10    0.00
## X7     0.00
## X6     0.00
## X8     0.00
```

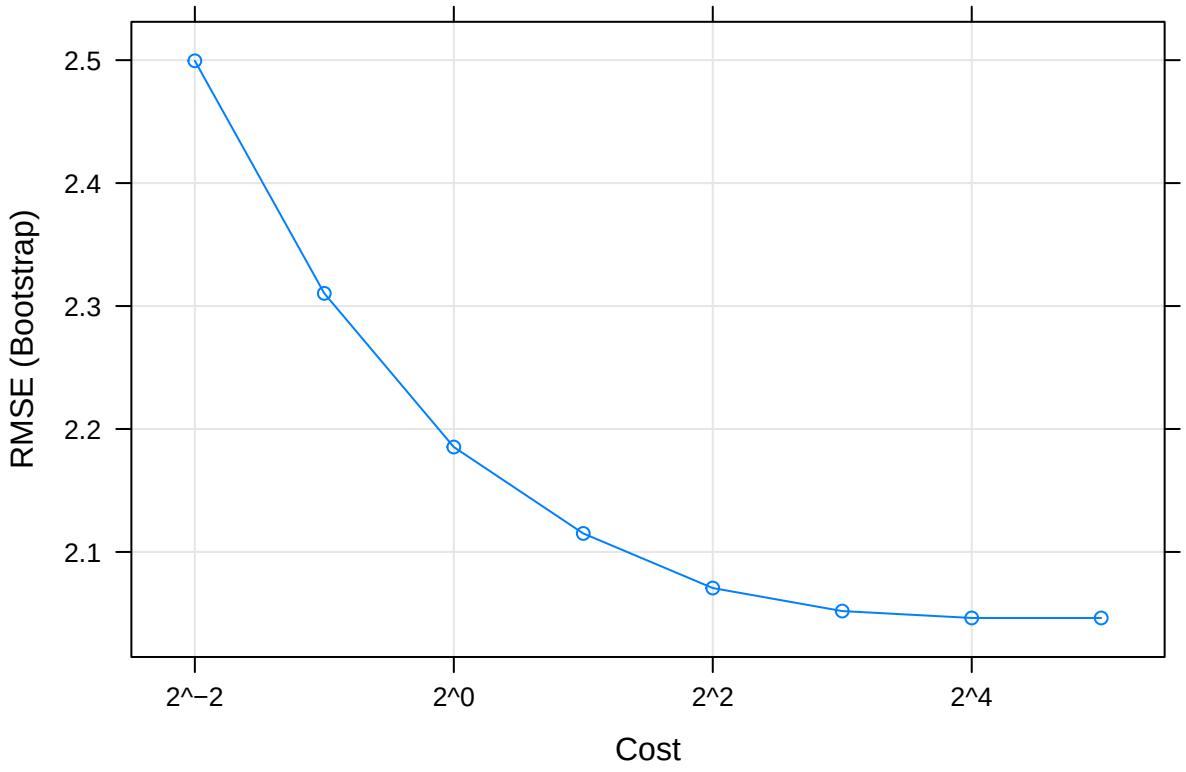
MARS only selects X1-X5 as important predictors.

**SVM**

```
set.seed(921)
svmRModel <- train(x = trainingData$x, y = trainingData$y, method = "svmRadial",
                   preProc = c("center", "scale"),
                   tuneLength = 8)
```

```
## Loading required package: kernlab
```

```
svmRPred <- predict(svmRModel, newdata = testData$x)

plot(svmRModel, scales = list(x = list(log = 2)))
```




```
postResample(pred = svmRPred, obs = testData$y)
```

```
##      RMSE  Rsquared
## 2.0667886 0.8267795
```

```
svmRModel$bestTune$C
```

```
## [1] 16
```

```
svmRModel$bestTune$sigma
```

```
## [1] 0.06046
```

```
round(getTrainPerf(svmRModel)[1, "TrainRMSE"], 2)
```

```
## [1] 2.05
```

The above plot shows that SVM tuning. The best model has a cost value of 16, sigma value of 0.06046, and an RMSE of 2.05%

Conlusions:

MARS model performs best followed by SVM and then K-NN. This makes sense because the data was not created using neighborhood information.

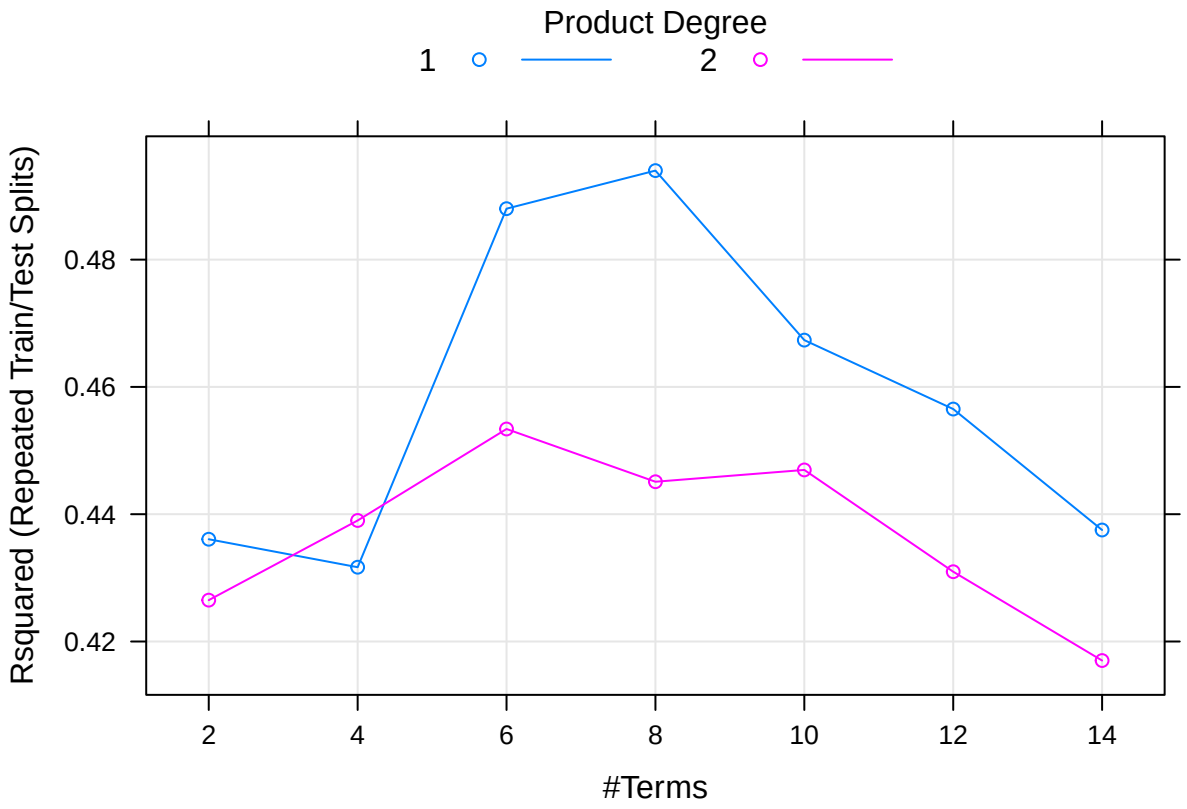
**KJ 7.4** Return to the permeability problem outlined in Exercise 6.2. Train several nonlinear regression models and evaluate the resampling and test set performance.

```
library(AppliedPredictiveModeling)
data(permeability)
#Identify and remove NZV predictors
nzvFingerprints = nearZeroVar(fingerprints)
noNzvFingerprints <- fingerprints[,-nzvFingerprints]
#Split data into training and test sets
set.seed(614)
trainingRows <- createDataPartition(permeability, p = 0.75, list = FALSE)

trainFingerprints <- noNzvFingerprints[trainingRows,]
trainPermeability <- permeability[trainingRows,]
testFingerprints <- noNzvFingerprints[-trainingRows,]
testPermeability <- permeability[-trainingRows,]

set.seed(614)
ctrl <- trainControl(method = "LGOCV")
```

a) Which nonlinear regression model gives the optimal resampling and test set performance?

**MARS**

```
marsPermGrid <- expand.grid(degree = 1:2, nprune = seq(2,14,by=2))
marsPermTune <- train(x = trainFingerprints, y = log10(trainPermeability), method = "earth", tuneGrid =

plot(marsPermTune,metric="Rsquared")
```



```
marsPermTune$results$degree[best(marsPermTune$results, "Rsquared", maximize = TRUE)]
```

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

```
marsPermTune$results$nprune[best(marsPermTune$results, "Rsquared", maximize = TRUE)]
```

```
## [1] 8
```

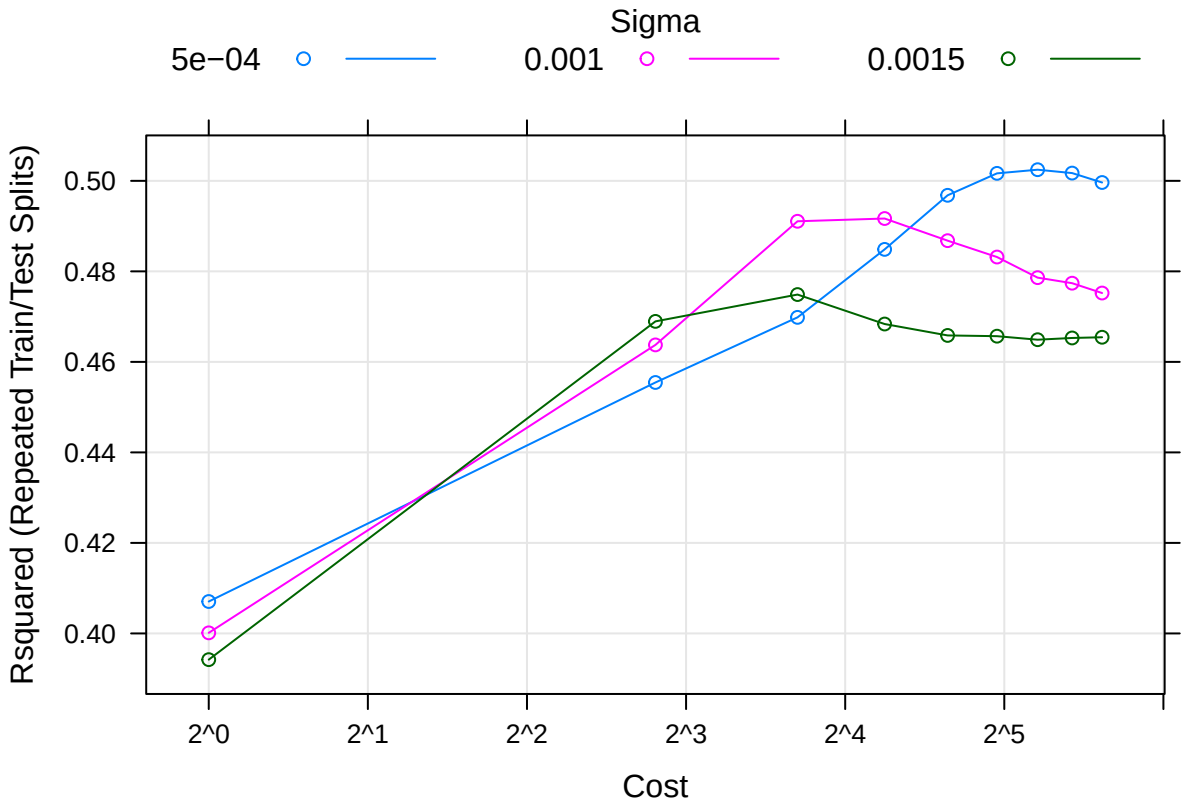
```
round(marsPermTune$results$Rsquared[best(marsPermTune$results, "Rsquared", maximize = TRUE)],2)
```

```
## [1] 0.49
```

The optimal degree that maxmize $R^2$ is 1 The optimal number of terms that maximize $R^2$ is 8 $R^2$ value is 0.49

**SVM**

```
SVMPermGrid <- expand.grid(sigma = c(0.0005,0.001,0.0015), C = seq(1,49,by=6))
SVMPermTune <- train(x = trainFingerprints, y = log10(trainPermeability),
                     method = "svmRadial",
                     trControl = ctrl,
                     tuneGrid = SVMPermGrid)

plot(SVMPermTune,metric="Rsquared", scales = list(x = list(log = 2)))
```



```
SVMPermTune$results$C[best(SVMPermTune$results, "Rsquared", maximize = TRUE)]
```

```
## [1] 37
```

```
SVMPermTune$results$sigma[best(SVMPermTune$results, "Rsquared", maximize = TRUE)]
```

```
## [1] 5e-04
```

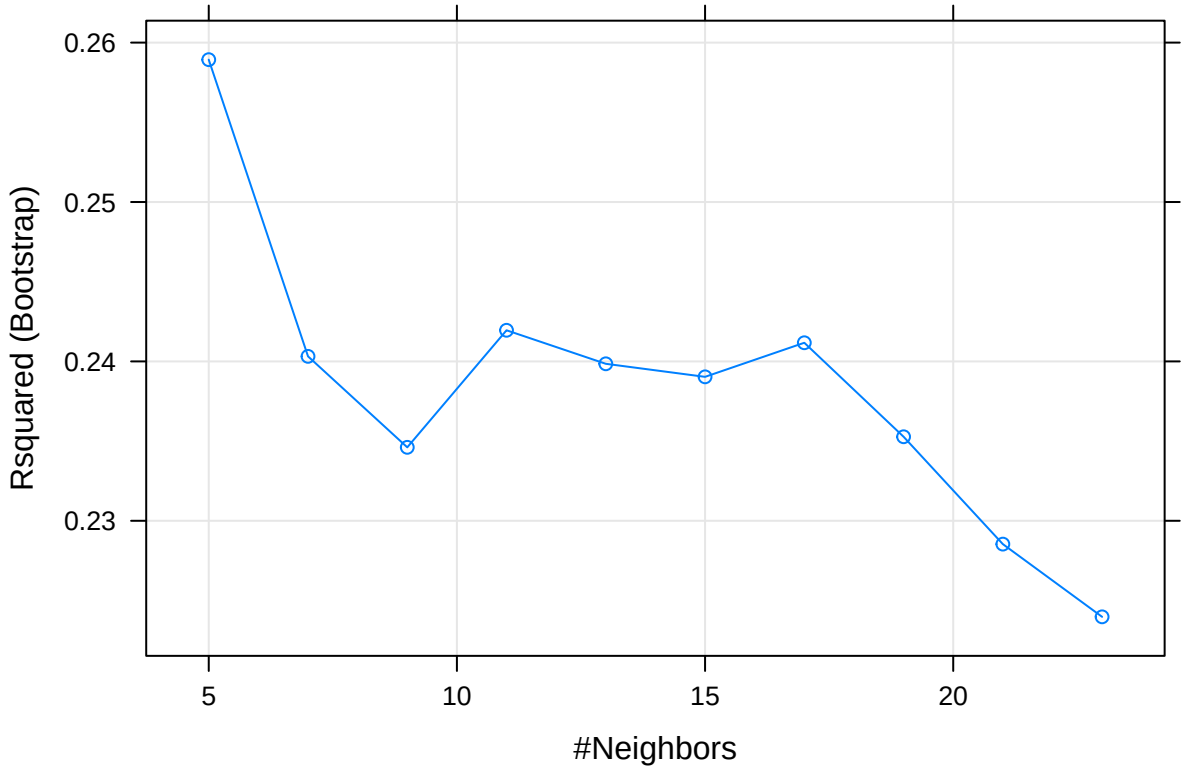
```
round(SVMPermTune$results$Rsquared[best(SVMPermTune$results, "Rsquared", maximize = TRUE)],2)
```

```
## [1] 0.5
```

The optimal cost that maxmize $R^2$ is 43 The optimal sigma maximize $R^2$ is 0.0005 $R^2$ value is 0.53

**KNN**

```
knnPermTune <- train(x = trainFingerprints, y = log10(trainPermeability), method = "knn", tuneLength =
plot(knnPermTune,metric="Rsquared")
```




```
knnPermTune$results$k[best(knnPermTune$results, "Rsquared", maximize = TRUE)]
```

```
## [1] 5
```

```
round(knnPermTune$results$Rsquared[best(knnPermTune$results, "Rsquared", maximize = TRUE)],2)
```

```
## [1] 0.26
```

The optimal number of nearest neighbors maxmize $R^2$ is 5 $R^2$ value is 0.21

b) Do any of the nonlinear models outperform the optimal linear model you previously developed in Exercise 6.2? If so, what might this tell you about the underlying relationship between the predictors and the response?

The radial SVM model performs best with a $R^2$ value of 0.53. This is not quite as good as the elastic net model. The results seem to show that there is a relationship between predictors and the responses and most likely best described by a linear structure.

c) Would you recommend any of the models you have developed to replace the permeability laboratory experiment?

I would not recommend any of the models to replace the permeability lab experiment, but if you had to replace it, replace it with the radiam SVM model.