# Test Question 2 (Models)

*Mario Ibanez*

*April 21, 2016*

## Introduction

Like in question 1 I will use R Markdown to make this report, which allows the code, plots, and commentary to all be included in a single PDF document. The first step is reading in the data and looking at some summary information:

```
# Read data and print summary information
Data2 <- read.csv("Data2.csv", header = TRUE)

# Summary and information
str(Data2)
```

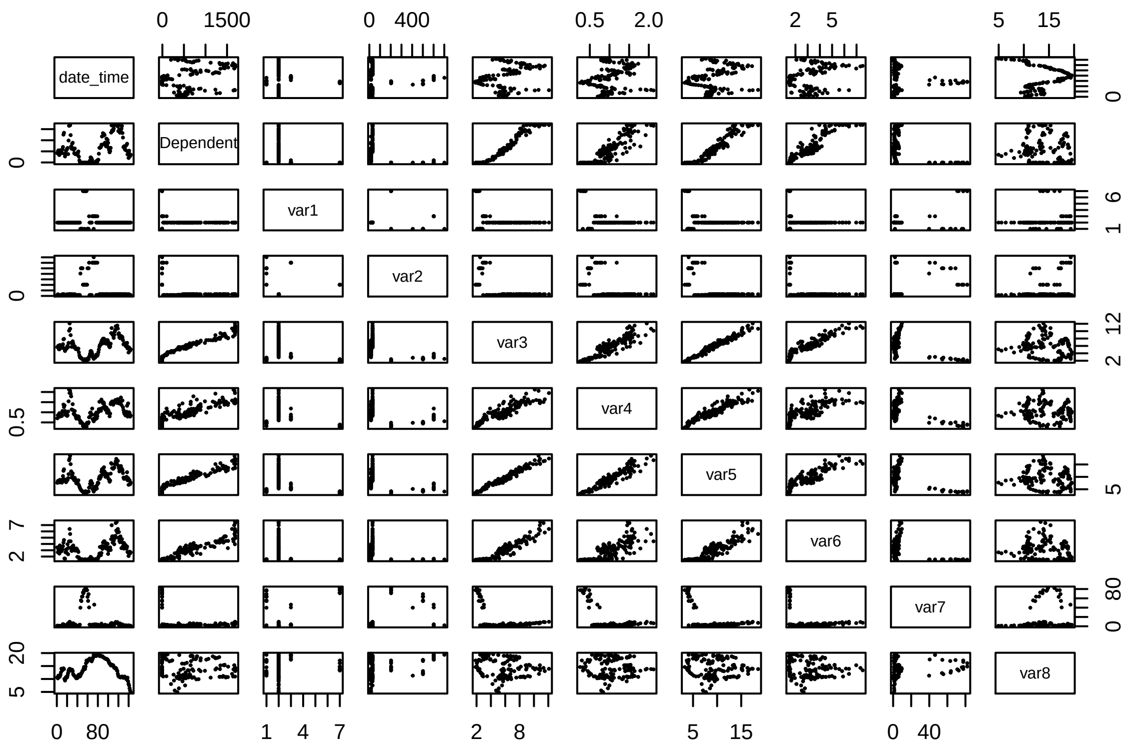
```
## 'data.frame':    144 obs. of  10 variables:
##  $ date_time: Factor w/ 144 levels "2015-10-11T07:00:00Z",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Dependent: num  415 494 542 353 334 ...
##  $ var1     : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ var2     : int  16 30 16 16 16 16 16 16 16 16 ...
##  $ var3     : num  5.68 5.81 6.07 5.12 5.08 ...
##  $ var4     : num  0.702 0.856 0.756 0.787 0.956 ...
##  $ var5     : num  7.59 8.33 8.43 8.14 7.77 ...
##  $ var6     : num  3.46 2.67 3.11 2.73 2.47 ...
##  $ var7     : num  1.431 1.177 0.934 0.956 1.067 ...
##  $ var8     : num  10.6 10.2 11.3 10.8 10.8 ...
```

```
summary(Data2)
```

```
##                  date_time     Dependent           var1
##  2015-10-11T07:00:00Z:  1   Min.   : -10.56   Min.   :1.000
##  2015-10-11T07:10:00Z:  1   1st Qu.: 191.61   1st Qu.:2.000
##  2015-10-11T07:20:00Z:  1   Median : 546.04   Median :2.000
##  2015-10-11T07:30:00Z:  1   Mean   : 651.17   Mean   :2.229
##  2015-10-11T07:40:00Z:  1   3rd Qu.:1031.44   3rd Qu.:2.000
##  2015-10-11T07:50:00Z:  1   Max.   :1690.44   Max.   :7.000
##  (Other)             :138
##       var2             var3              var4              var5
##  Min.   : 16.0   Min.   : 1.871   Min.   :0.2556   Min.   : 3.282
##  1st Qu.: 16.0   1st Qu.: 4.313   1st Qu.:0.7883   1st Qu.: 7.302
##  Median : 30.0   Median : 5.813   Median :0.9818   Median : 9.008
##  Mean   :101.8   Mean   : 5.940   Mean   :1.0434   Mean   : 9.406
##  3rd Qu.: 30.0   3rd Qu.: 7.519   3rd Qu.:1.3861   3rd Qu.:11.896
##  Max.   :701.0   Max.   :12.093   Max.   :2.1203   Max.   :18.505
##
##       var6             var7              var8
##  Min.   :1.500   Min.   : 0.8757   Min.   : 4.956
##  1st Qu.:1.697   1st Qu.: 1.2977   1st Qu.:10.916
##  Median :3.082   Median : 2.5374   Median :13.685
##  Mean   :3.118   Mean   :11.1436   Mean   :13.902
##  3rd Qu.:3.783   3rd Qu.: 6.0091   3rd Qu.:17.217
##  Max.   :7.502   Max.   :82.0001   Max.   :19.544
##
```

It is also good to plot the data:

```
plot(Data2, pch = 20, cex = 0.3)
```



This plot tells us a lot. It appears as if var1 and var2 may be categorical variables even though they appear numeric. The code below allows us to take a closer look at these variables:

```
table(Data2$var1)
```

```
##
##   1   2   3   7
##  11 117   9   7
```

```
table(Data2$var2)
```

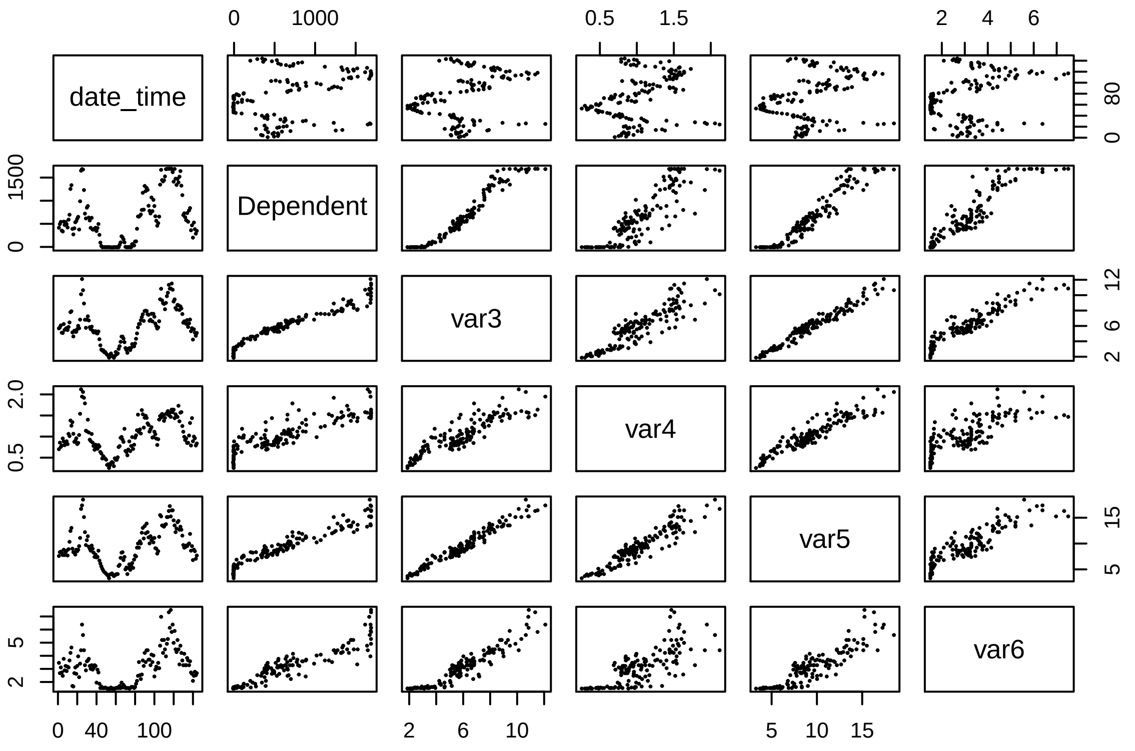
```
##
##  16  30 203 406 501 601 602 701
##  51  66   9   1   7   1   8   1
```

```
table(Data2$var1, Data2$var2)
```

```
##
##      16  30 203 406 501 601 602 701
##    1  0   0   2   1   7   0   0   1
##    2 51  66   0   0   0   0   0   0
##    3  0   0   0   0   0   1   8   0
##    7  0   0   7   0   0   0   0   0
```

var1 only has 4 different values in 144 observations and the majority of the values are 2. var2 is also an unusual variable. I suspect that these variables are coming from something other than sensors and that they may be categorical. In the first column of the plot above it's also worth noting that var3, var4, var5, and var6 closely follow the behavior of the dependent variable as time passes. However, var3, var4, var5, and var6 are also highly correlated with each other. This can be seen in multiple plots above, though here is a zoomed in version for just these variables as well as a correlation matrix:

```
# Plot and correlation matrix for var3, var4, var5, and var6
plot(Data2[, c(1, 2, 5, 6, 7, 8)], pch = 20, cex = 0.3)
```



```
cor(Data2[, c(2, 5, 6, 7, 8)])
```

```
##             Dependent      var3      var4      var5      var6
## Dependent 1.0000000 0.9678761 0.8682951 0.9574208 0.9249466
## var3      0.9678761 1.0000000 0.8874766 0.9788503 0.9406772
## var4      0.8682951 0.8874766 1.0000000 0.9356354 0.7502589
## var5      0.9574208 0.9788503 0.9356354 1.0000000 0.8961213
## var6      0.9249466 0.9406772 0.7502589 0.8961213 1.0000000
```

var3 and var5 are especially correlated with each other. These two variables are also highly correlated with the dependent variable.

## Method 1 (Simple linear regression)

One approach is to ignore the fact that this is time series data. In other words, this can be thought of as a static time series regression model. It is assumed that the independent variables instantaneously affect (or change with) the dependent variable (there is no lag). For example, if we know the values of all the independent variables at some instant, we can predict or have a very good idea about the value of the dependent variable. Since the goal seems to be predict or model the behavior of the dependent variable based on the known variables, this approach is reasonable.

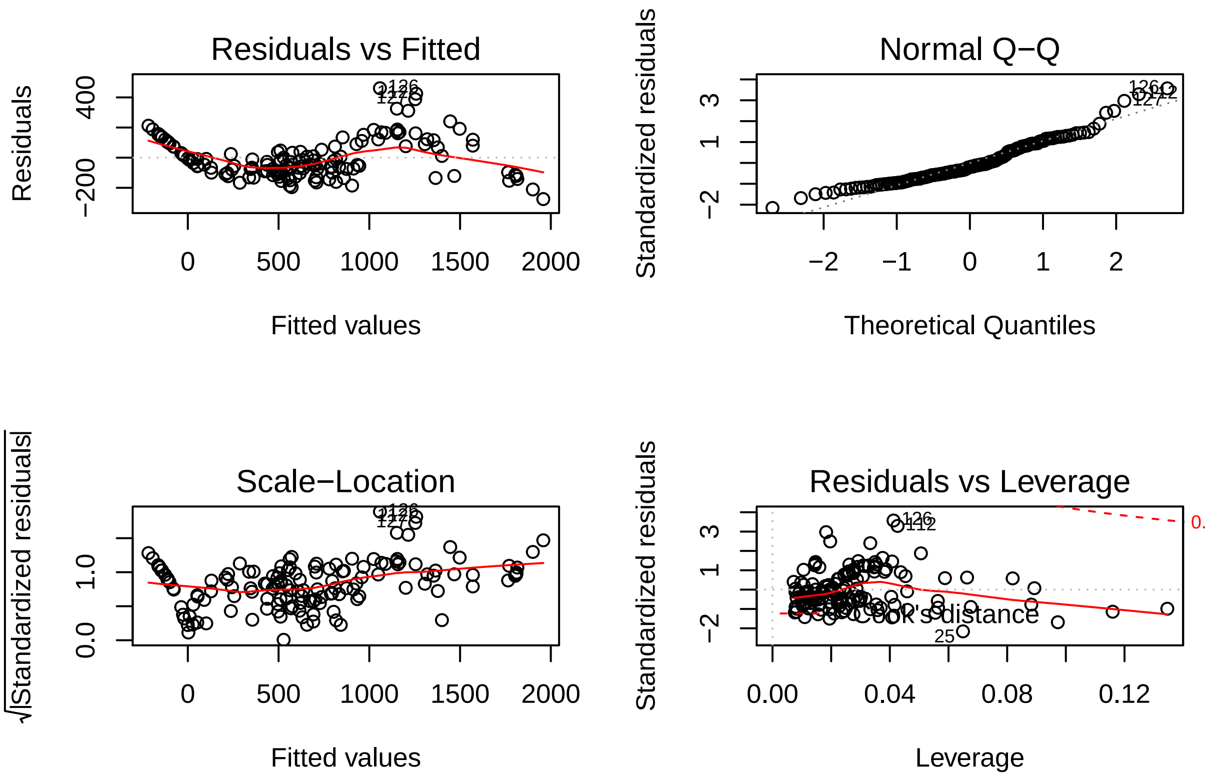
We can start with a simple and naive approach, fitting the model with var3, var5, and var6 since these are very highly correlated to the dependent variable.

```
ols_fit <- lm(Dependent ~ var3 + var5 + var6, data = Data2); summary(ols_fit)
```

```
##
## Call:
## lm(formula = Dependent ~ var3 + var5 + var6, data = Data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -274.83  -93.88  -26.31   90.97  460.77
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -703.40      32.45 -21.675  < 2e-16 ***
## var3           97.23      31.41   3.096 0.002372 **
## var5           55.79      16.37   3.409 0.000852 ***
## var6           80.87      25.31   3.195 0.001730 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 131.7 on 140 degrees of freedom
## Multiple R-squared:  0.9433, Adjusted R-squared:  0.9421
## F-statistic: 776.6 on 3 and 140 DF,  p-value: < 2.2e-16
```

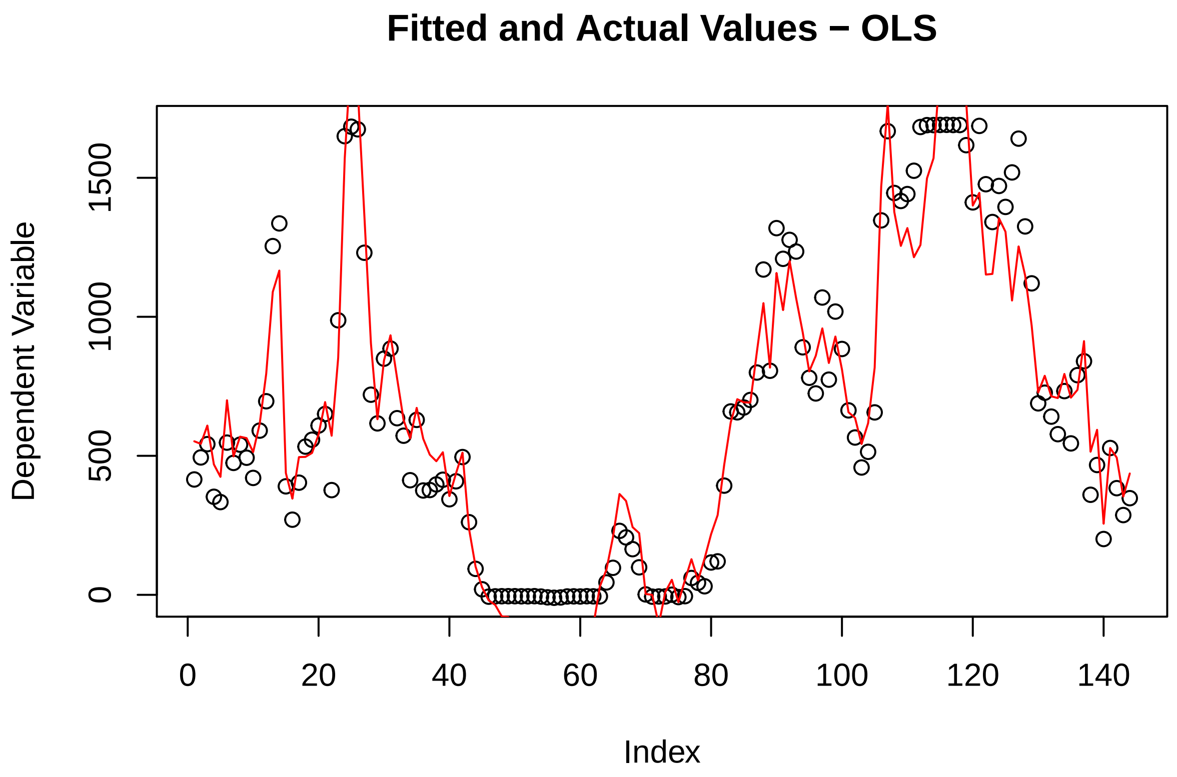
The $R^2$ value is 0.9433 which means that the variables var3, var5, and var6 account for 94.33% of the variation in the dependent variable. Some diagnostic plots can be plotted with the command below:

```
par(mfrow = c(2, 2))
plot(ols_fit); par(mfrow = c(1, 1))
```



Overall the diagnostics do not look bad. There is a bit of a patter in the first plot, Residuals vs Fitted. We can also look at a plot of the dependent variable and the fitted values to see how close they line up:

```r
plot(Data2$Dependent, ylab = "Dependent Variable",
     main = "Fitted and Actual Values - OLS")
points(ols_fit$fitted.values, type = "l", col = "red")
```



Even though it was a simple model, the fit is pretty good.

## Method 2 (Random Forest)

For the next 3 methods, I will use the *caret* package in R. To compare the next three models, I will split the data into a training set and test set. First, the *caret* package needs to be loaded:

```r
# Load package and set seed
library(caret)
set.seed(1234)

train_index <- createDataPartition(y = Data2$Dependent,
                                   times = 1, p = 0.80, list = FALSE)
Data2_train <- Data2[train_index, ]
Data2_test <- Data2[-train_index, ]
```

The training set is 116 observations out of 144 and the test set is 28 observations.

Next the package *randomForest* needs to be loaded and we can go ahead and fit the model:

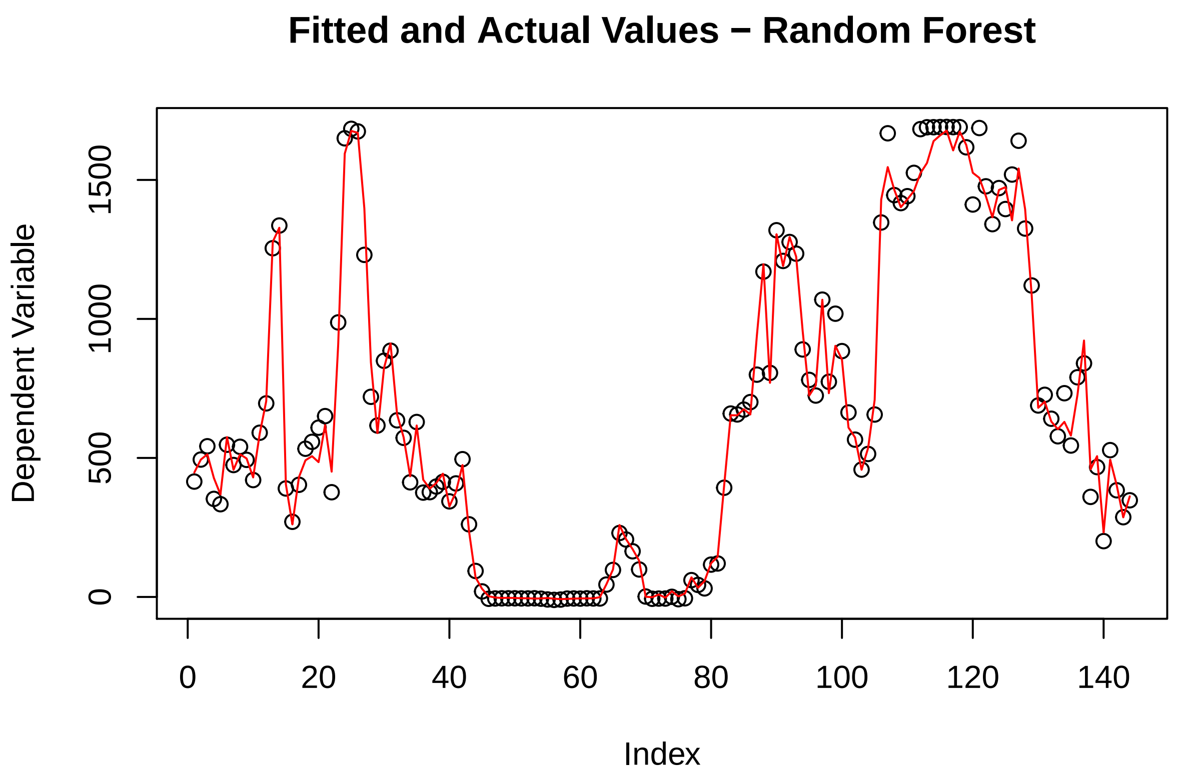
```r
library(randomForest)

forest_model <- randomForest(Dependent ~ var3 + var4 + var5 + var6 + var7 + var8,
                             data = Data2_train,
                             xtest = Data2_test[ , 5:10],
                             ytest = Data2_test[, 2], keep.forest = TRUE)
forest_model
```

```
##
## Call:
##  randomForest(formula = Dependent ~ var3 + var4 + var5 + var6 +      var7 + var8, data = Data2_train
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 8527.773
##                     % Var explained: 97.01
##                        Test set MSE: 6283.02
##                     % Var explained: 98.24
```

I chose to train the random forest on all the variables besides var1 and var2. 98.2% of the variation in the test set was explained by the model. The MSE on the test set was 6349.08. Like I did above, the predicted values can be plotted along with the observed values so we can visualize how close the fit was.

```r
plot(Data2$Dependent, ylab = "Dependent Variable",
     main = "Fitted and Actual Values - Random Forest")
points(predict(forest_model, newdata = Data2[, 5:10]), type = "l", col = "red")
```



The red line is the predicted values, and we see that they follow the observed values very closely.

## Method 3 (Regression with stepwise selection)

The next model will be a regression model incorporating step-wise selection. This process can take a long time by hand but is very quick using the *train()* function with the method *lmStepAIC* chosen. I will include all 8 independent variables since the algorithm will search for the one with the lowest AIC automatically.

```
step_model <- train(Dependent ~ var1 + var2 + var3 + var4 + var5 + var6 + var7 + var8,
                        data = Data2_train, method = "lmStepAIC")
```

```
step_model
```

```
## Linear Regression with Stepwise Selection
##
## 116 samples
##   9 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 116, 116, 116, 116, 116, 116, ...
## Resampling results:
##
##   RMSE     Rsquared
##   147.164  0.9293967
##
##
```

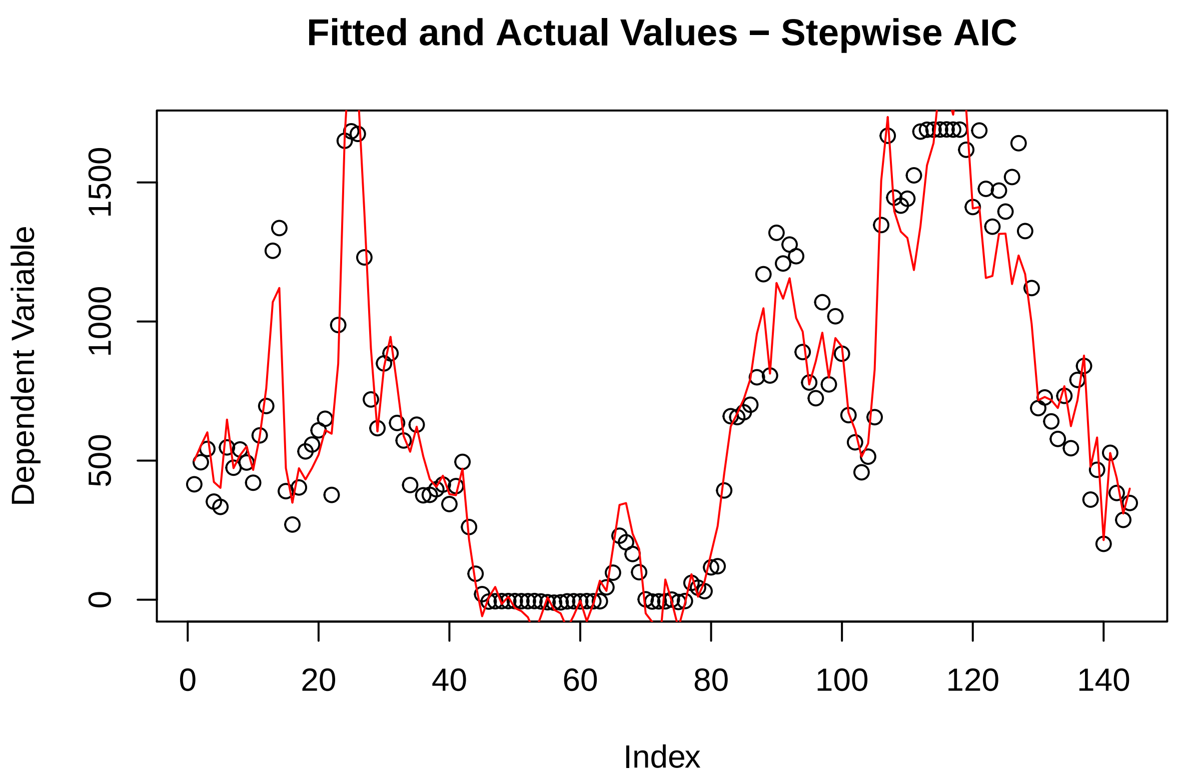
Now to get the MSE on the test set, we can do that by hand in R:

```
# Calculates the MSE on the test set
mean((predict(step_model, newdata = Data2_test[, 3:10]) - Data2_test$Dependent)^2)
```

```
## [1] 8206.09
```

The MSE was 8206.08961, higher than the previous method. As before it's interesting to compare the predicted values to the observed values:

```
plot(Data2$Dependent, ylab = "Dependent Variable",
     main = "Fitted and Actual Values - Stepwise AIC")
points(predict(step_model, newdata = Data2[, 3:10]), type = "l", col = "red")
```



Compared with the previous corresponding plot, we can see that the red line (predicted values) does not follow the observed data quite as closely.

## Method 4 (Regression with PCA)

The final method will again be from the *caret* package, this time regression with principal component analysis which acts to reduce the the number of variables modeling the dependent variable. This is especially useful in this case because we have very high correlation between some of the independent variables.

```
pca_model <- train(Dependent ~ var1 + var2 + var3 + var4 + var5 + var6 + var7 + var8,
                   data = Data2_train, method = "pcr")
```

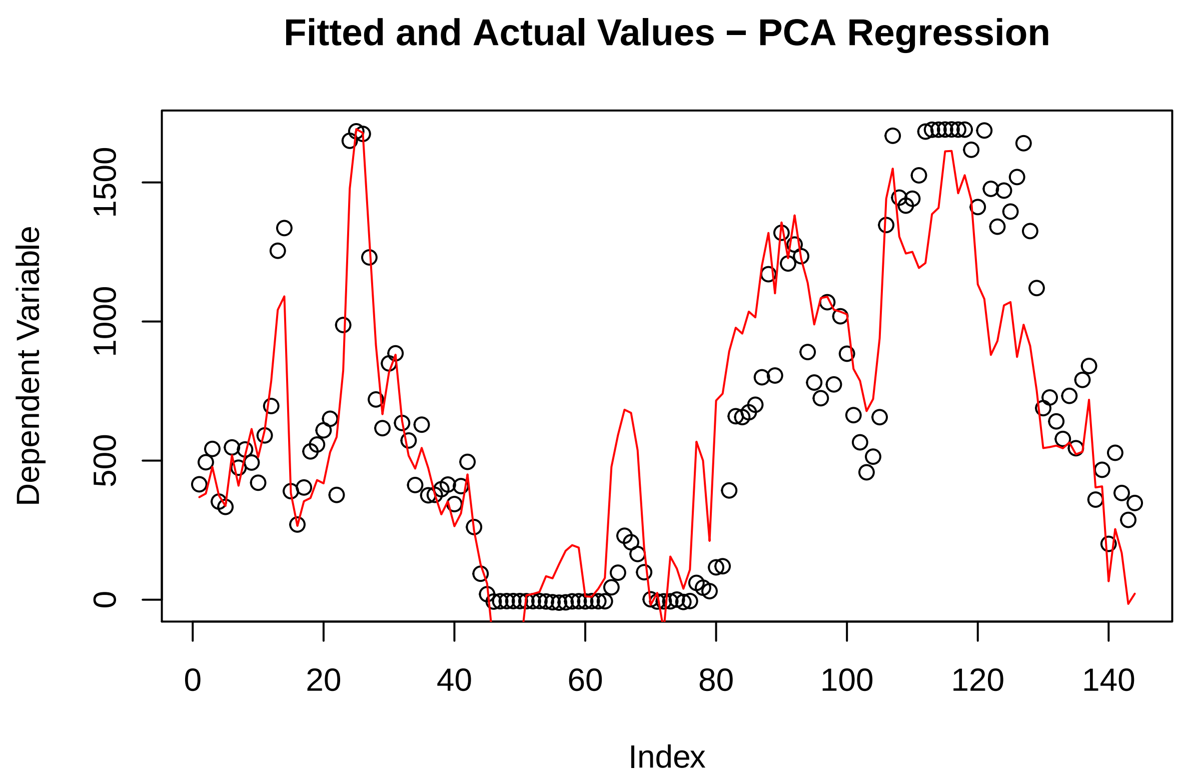
In order to get the MSE for the test set predictions, we can do it by hand again in R as was done earlier:

```
mean((predict(pca_model, newdata = Data2_test[, 3:10]) - Data2_test$Dependent)^2)
```

```
## [1] 50215.86
```

The MSE on the test set is 50215.85901. We can also look at the plot of all predicted values compared to the observed values as was done in earlier methods:

```
plot(Data2$Dependent, ylab = "Dependent Variable",
     main = "Fitted and Actual Values - PCA Regression")
points(predict(pca_model, newdata = Data2[, 3:10]), type = "l", col = "red")
```

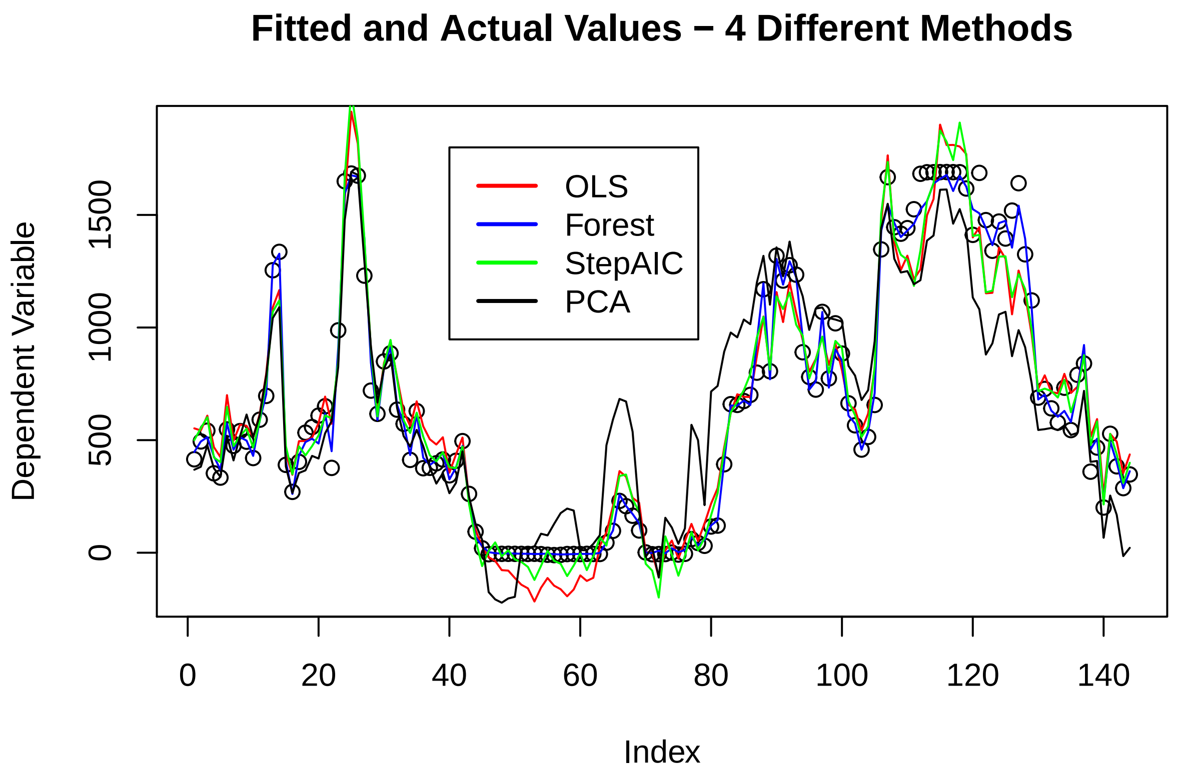


Visually it's apparent that this method did not provide a good fit.

## Conclusion

To summarize the findings of the 4 methods, below is a plot of the predicted values of each method all in a single plot:

```r
plot(Data2$Dependent, ylab = "Dependent Variable",
     main = "Fitted and Actual Values - 4 Different Methods",
     ylim = c(-200, 1900))
points(ols_fit$fitted.values, type = "l", col = "red")
points(predict(forest_model, newdata = Data2[, 5:10]), type = "l", col = "blue")
points(predict(step_model, newdata = Data2[, 3:10]), type = "l", col = "green")
points(predict(pca_model, newdata = Data2[, 3:10]), type = "l", col = "black")
legend(40, 1800,
       col = c("red", "blue", "green", "black"),
       legend = c("OLS", "Forest", "StepAIC", "PCA"), lwd = 2)
```



Visually, the random forest model very closely modeled the behavior of the dependent variable. The table below summarizes the results of the four methods:

| Method | Test Set MSE | Overall MSE | # of Variables |
|---|---|---|---|
| Ordinary Least Squares | N/A | 16872.21 | 3 |
| Random Forest | 6349.08 | 2581.91 | 6 |
| Stepwise AIC Regression | 8206.09 | 14341.73 | 8 |
| PCA Regression | 50215.86 | 57616.43 | 8 |

Note, the MSE for the ordinary least squares case was calculated the same way that the other overall MSE values were calculated. 16872.21 is not the traditional unbiased estimator of $\sigma^2$ but rather the literal mean of the squared errors.

Based on the plots and this table, it would appear that the random forest model performed the best of the four in the report. It is not surprising that the fits were so good as we saw how similar the curves were in the plots at the beginning of this report. The worst performing of the four was PCA Regression. It would be my belief that the dependent variable was a reading from a device similar to the readings from whatever devices var3, var4, var5 and var6 are. This could mean that the dependent variable is a reading from a device that is possibly redundant, or perhaps they are all redundant sensors and all are used in case one happens to malfunction.

In the end, I decided to ignore the fact that this is time series data. What this assumption means is that the predictions can be made only instantaneously. If the values of the independent variables are known at some moment, then the models predict the value of the dependent variable at that same moment.