Exercises for *Applied Predictive Modeling* Chapter 6 — Linear Regression and Its Cousins

Max Kuhn, Kjell Johnson

Version 1
November 12, 2014

The solutions in this file uses several R packages not used in the text. To install all of the packages needed for this document, use:

```
> install.packages(c("AppliedPredictiveModeling", "caret", "elasticnet", "pls",
+                    "RColorBrewer", "reshape2"))
```

Exercise 1

Infrared (IR) spectroscopy technology can be used to determine the chemical makeup of a substance. The theory of IR spectroscopy holds that unique molecular structures absorb IR frequencies differently. In practice a spectrometer fires a series of IR frequencies into a sample material, and the device measures the absorbance of the sample at each individual frequency. This series of measurements creates a spectrum profile which can then be used to determine the chemical make–up of the sample material.

A Tecator Infratec Food and Feed Analyzer instrument was used to analyze 215 samples of meat across 100 frequencies. A sample of these frequency profiles are displayed in Figure 1. In addition an IR profile, analytical chemistry was used to determine the percent content of water, fat, and protein for each sample. If we can establish a predictive relationship between IR spectrum and fat content, then food scientists could predict a sample's fat content instead of using analytical chemistry to determine the content. This would provide costs savings, since analytical chemistry is a more expensive, time consuming process.

(a) Start R and use these commands to load the data:

```
> library(caret)
> data(tecator)
> # use ?tecator to see more details
```

The matrix `absorp` contains the 100 absorbance values for the 215 samples, while matrix `endpoints` contains the percent of moisture, fat, and protein in columns 1–3, respectively.

(b) In this example the predictors are the measurements at the individual frequencies. Because the frequencies lie in a systematic order (850 to 1050 nm), the predictors have a high degree of correlation. Hence, the data lies in a smaller dimension than the total number of predictors (215). Use PCA to determine the effective dimension of this data. What is the effective dimension?

(c) Split the data into a training and a test set, pre–process the data, and build each variety of model described in this chapter. For those models with tuning parameters, what are the optimal values of the tuning parameter(s)?

(d) Which model has the best predictive ability? Is any model significantly better or worse than the others?

(e) Explain which model you would use for predicting the fat content of a sample.

Solutions

The full set of principal components can be obtained using the `prcomp` function:

```
> pcaObj <- prcomp(absorp, center = TRUE, scale = TRUE)
```

To get the percent of variance associated with each component, we can get the standard deviation object:

```
> pctVar <- pcaObj$sdev^2/sum(pcaObj$sdev^2)*100
> head(pctVar)
```

```
[1] 98.626193  0.969705  0.279324  0.114430  0.006461  0.002625
```

This indicates that the first components accounts for almost all of the information in the data. Based on this analysis, the true dimensionality is much lower than the number of predictors. However, this is based on a *linear* combination of the data; other nonlinear summarizations of the predictors may also be useful.

There are three endpoints and we will model the third, which is the percentage of protein. Given the sample size, 25% of the data will be held back for training and five repeats of 10–fold cross–validation will be used to tune the models:

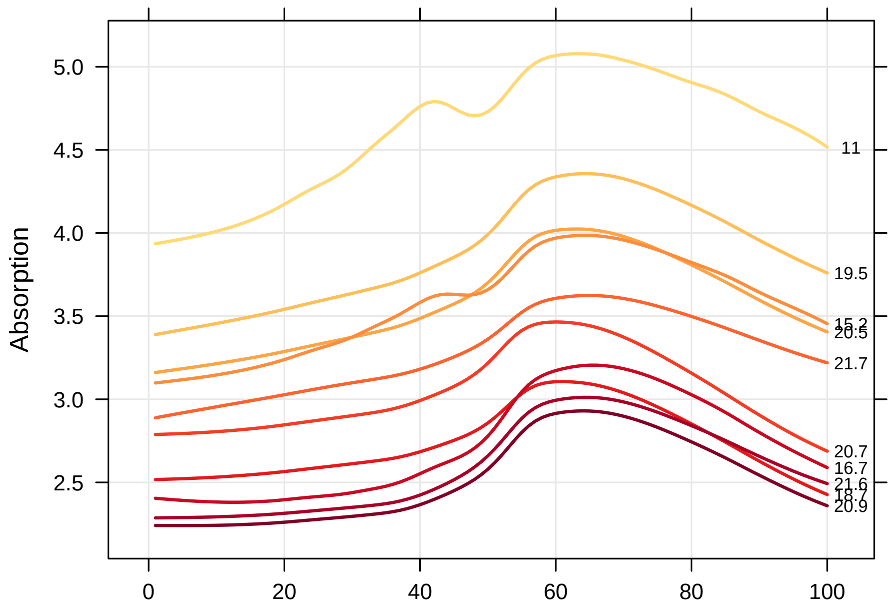

Figure 1: A sample of 10 spectrum of the Tecator data. The colors of the curves reflect the absorption values, where yellow indicates low absorption and red is indicative of high absorption.

```
> set.seed(1029)
> inMeatTraining <- createDataPartition(endpoints[, 3], p = 3/4, list= FALSE)
>
> absorpTrain <- absorp[ inMeatTraining,]
> absorpTest  <- absorp[-inMeatTraining,]
> proteinTrain <- endpoints[ inMeatTraining, 3]
> proteinTest  <- endpoints[-inMeatTraining,3]
>
> ctrl <- trainControl(method = "repeatedcv", repeats = 5)
```

To start, a simple linear model was used for these data:

```
> set.seed(529)
> meatLm <- train(x = absorpTrain, y = proteinTrain, method = "lm", trControl = ctrl)
> meatLm

Linear Regression

163 samples
100 predictors
```

```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)

Summary of sample sizes: 147, 146, 147, 146, 147, 146, ...

Resampling results

  RMSE   Rsquared  RMSE SD  Rsquared SD
  1.382  0.8716    1.014    0.08729
```

The RMSE is slightly more than 1%. However, it is a fair to assume that the amount of multicolinearity in the predictors is probably degrading performance. For this reason, PCR and PLS models are also trained:

```
> set.seed(529)
> meatPCR <- train(x = absorpTrain, y = proteinTrain,
+                  method = "pcr",
+                  trControl = ctrl, tuneLength = 25)
```

```
> set.seed(529)
> meatPLS <- train(x = absorpTrain, y = proteinTrain,
+                  method = "pls",
+                  trControl = ctrl,
+                  preProcess = c("center", "scale"),
+                  tuneLength = 25)
> ## For Figure
> comps <- rbind(meatPLS$results, meatPCR$results)
> comps$Model <- rep(c("PLS", "PCR"), each = 25)
```

The results are shown in Figure 2. Both models achieve comparable error rates but the PLS model requires fewer (17) components. The RMSE for the PLS model was estimated to be 0.65%.

An elastic net model was also tuned over five values of the L_2 regularization parameter and the L_1 regularization parameter:

```
> set.seed(529)
> meatENet <- train(x = absorpTrain, y = proteinTrain,
+                   method = "enet",
+                   trControl = ctrl,
+                   preProcess = c("center", "scale"),
+                   tuneGrid = expand.grid(lambda = c(0, .001, .01, .1, 1),
+                                          fraction = seq(0.05, 1, length = 20)))
```

The final model used $\lambda = 0$ and a fraction of 0.1, which corresponds to retaining 0 of the 100 predictors. This parameter combination corresponds to a Lasso model. Figure 3 shows a heat map

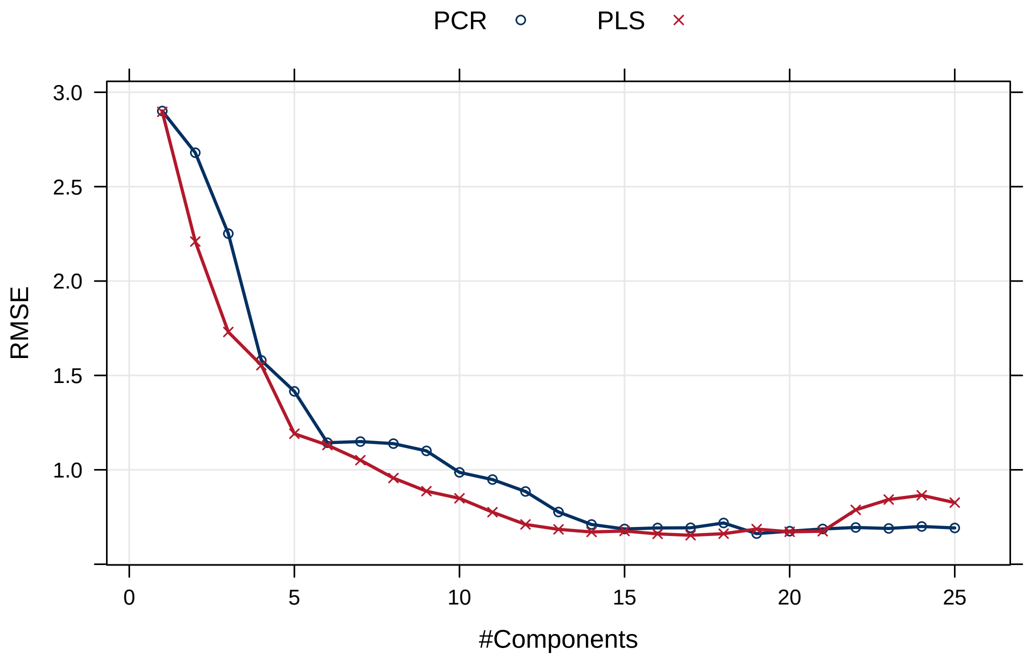

Figure 2: The resampling performance profile for the PCR and PLS models built using the Tecator protein data.

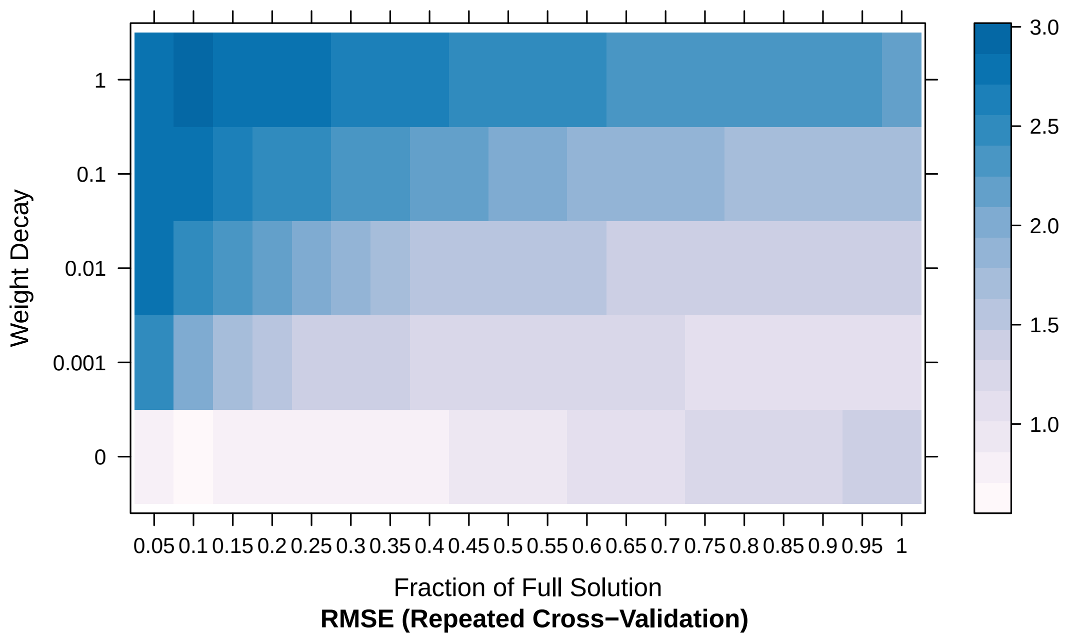

Figure 3: The resampling performance profile for the elastic net model built using the Tecator protein data.

of the resampled RMSE values across the two tuning parameters. The RMSE associated with the optimal parameters was 0.7%.

Based solely on the resampling statistics, the PLS model would probably be preferred over the Lasso model since it is especially suited to handling highly correlated data. The ordinary linear model, as expected, had the worse performance overall.

Exercise 2

Developing a model to predict permeability (see Sect. 1.4) could save significant resources for a pharmaceutical company, while at the same time more rapidly identifying molecules that have a sufficient permeability to become a drug.

(a) Start R and use these commands to load the data:

```
> library(AppliedPredictiveModeling)
> data(permeability)
```

The matrix `fingerprints` contains the 1107 binary molecular predictors for the 165 compounds, while `permeability` contains permeability response.

(b) The fingerprint predictors indicate the presence or absence of substructures of a molecule and are often sparse meaning that relatively few of the molecules contain each substructure. Filter out the predictors that have low frequencies using the `nearZeroVar` function from the caret package. How many predictors are left for modeling?

(c) Split the data into a training and a test set, pre–process the data, and tune a partial least squares model. How many latent variables are optimal and what is the corresponding resampled estimate of R^2?

(d) Predict the response for the test set. What is the test set estimate of R^2?

(e) Try building other models discussed in this chapter. Do any have better predictive performance?

(f) Would you recommend any of your models to replace the permeability laboratory experiment?

Solutions

Before pre-processing and modeling the data, let's examine the distribution of the permeability response (left-hand plot in Figure 4). Clearly the distribution is not symmetric, but is skewed-to-the-right. When the response has this kind of distribution, an appropriate next step is to log-transform the data. We then should verify if the transformation induced a symmetric distribution (right-hand plot in Figure 4). For this data, the log-transformation did induce an approximately symmetric distribution. For this problem, we will model the log-transformed response.

Next, let's pre-process the fingerprint predictors to determine how many predictors will be removed and how many will remain for modeling. The following syntax can be used to identify and remove near-zero variance fingerprint predictors.

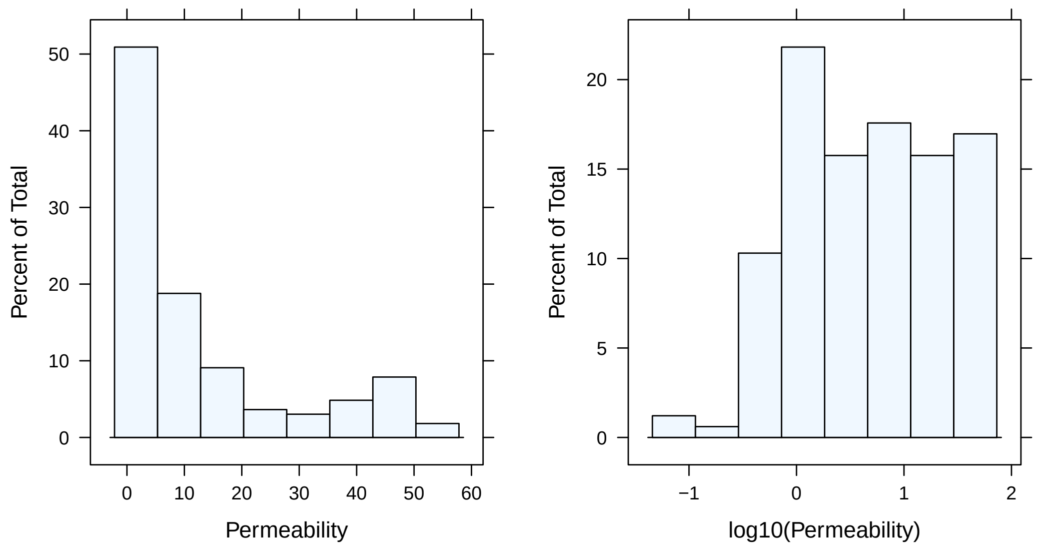

Figure 4: The response distribution for the permeability data.

```
> #Identify and remove NZV predictors
> nzvFingerprints = nearZeroVar(fingerprints)
> noNzvFingerprints <- fingerprints[,-nzvFingerprints]
```

In total there are 719 near-zero variance fingerprints, leaving 388 left for modeling. This is a significant reduction from the original matrix, and indicates that many of the fingerprints are describing unique features of very small subsets of molecules. Because this data set has a small number of samples relative to predictors (n=165 vs p=388), we will hold out 25% in the test set for evaluating model performance.

```
> #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")
>
> plsTune <- train(x = trainFingerprints, y = log10(trainPermeability),
+                  method = "pls",
+                  tuneGrid = expand.grid(ncomp = 1:15),
+                  trControl = ctrl)
```

Figure 5 indicates that the optimal number of latent variables that maximizes R^2 is 9.

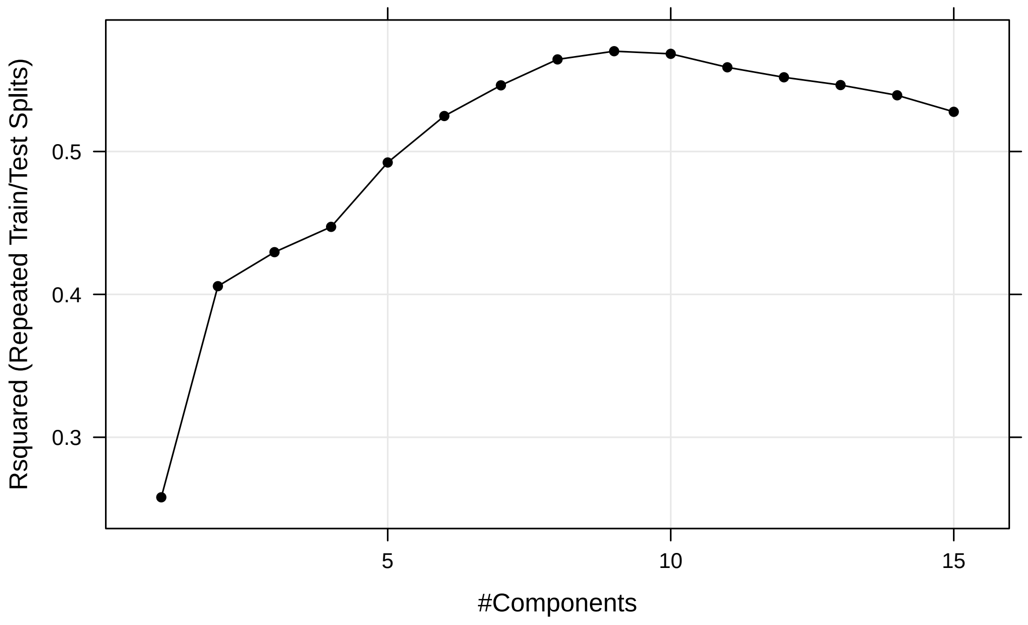

Figure 5: PLS tuning parameter profile for the permeability data

The relationship between the PLS components and the response can be seen in Figure 6, where the straight line is the linear regression fit, and the curved line is the loess fit. For the training data, the first three components are linearly related to log10(permeability). Each component also contains a noticeable amount of noise relative to predicting the response.

Next we will predict the response for the test set and compare the R^2 value with the one obtained through leave-group-out cross-validation. The relationship is displayed in Figure 7, where the R^2 value is 0.541. Leave-group-out cross-validation estimated that the PLS model would have an R^2 value of 0.57. These values are close, indicating that the leave-group-out cross-validation performance is a good indicator of the performance from a true hold-out set.

In addition to a PLS model, ridge regression and elastic net models were tuned. The tuning parameter profiles can be seen in Figures 8 and 9. For ridge regression, the optimal R^2 value

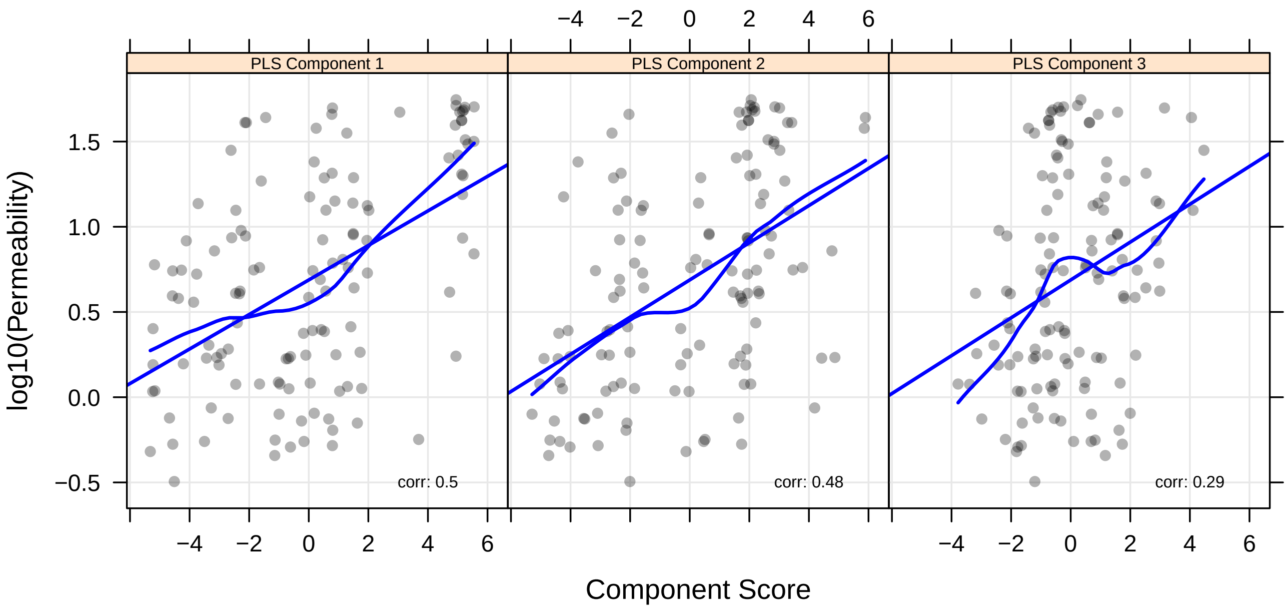

Figure 6: PLS scores versus log10(permeability).

is 0.561, which is obtained at a weight decay value of 0.14375. This shrinkage parameter value is relatively high and drives the coefficients more rapidly towards zero. For the elastic net, the optimal R^2 value is 0.58, which is obtained with the fraction of the full solution set at 0.1688 and a weight decay of 0.1. Therefore, the fraction parameter lessens the weight decay, relative to ridge regression. Clearly, the elastic net solution the best penalized model choice for this problem.

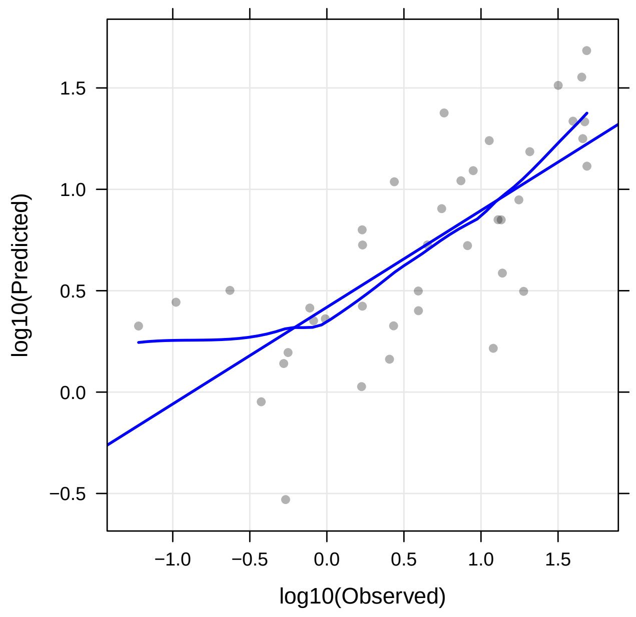

Figure 7: PLS predictions for the test set for the permeability data.

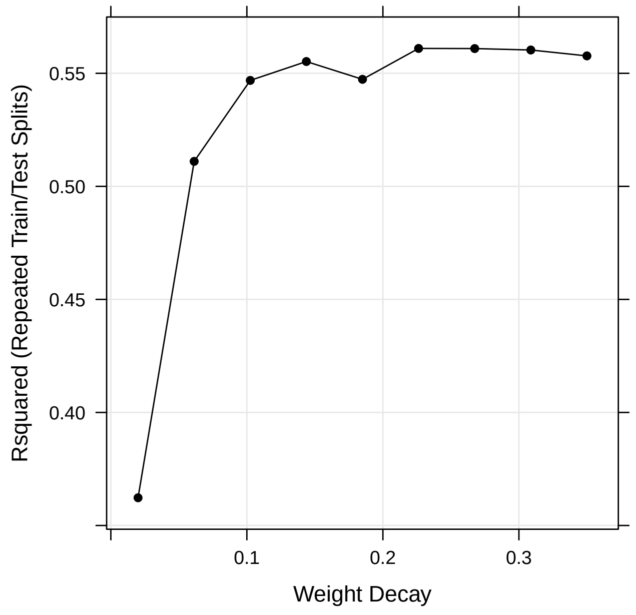

Figure 8: Ridge regression tuning parameter profile for the permeability data.

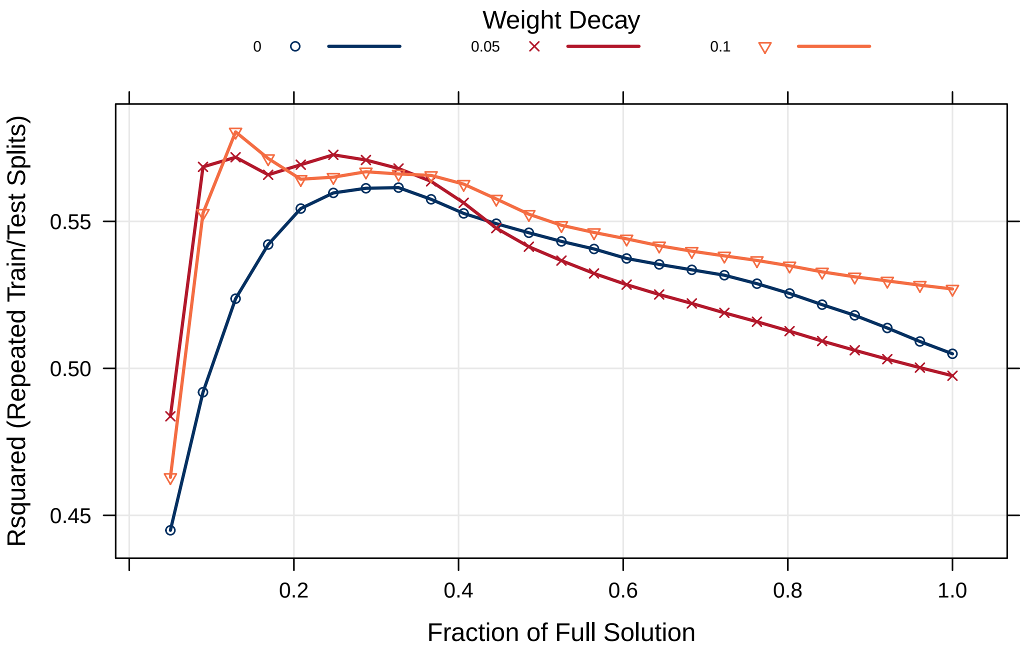

Figure 9: Elastic net tuning parameter profile for the permeability data.

Exercise 3

A chemical manufacturing process for a pharmaceutical product was discussed in Sect. 1.4. In this problem, the objective is to understand the relationship between biological measurements of the raw materials (predictors), measurements of the manufacturing process (predictors) and the response of product yield. Biological predictors cannot be changed but can be used to assess the quality of the raw material before processing. On the other hand, manufacturing process predictors can be changed in the manufacturing process. Improving product yield by 1% will boost revenue by approximately one hundred thousand dollars per batch.

(a) Start R and use these commands to load the data:

```
> library(AppliedPredictiveModeling)
> data(ChemicalManufacturingProcess)
```

The matrix `processPredictors` contains the 57 predictors (12 describing the input biological material, and 45 describing the process predictors) for the 176 manufacturing runs. `yield` contains the percent yield for each run.

(b) A small percentage of cells in the predictor set contain missing values. Use an imputation function to fill in these missing values (e.g. see Sect. 3.8).

(c) Split the data into a training and a test set, pre–process the data, and tune a model of your choice from this chapter. What is the optimal value of the performance metric?

(d) Predict the response for the test set. What is the value of the performance metric and how does this compare with the resampled performance metric on the training set?

(e) Which predictors are most important in the model you have trained? Do either the biological or process predictors dominate the list?

(f) Explore the relationships between each of the top predictors and the response. How could this information be helpful in improving yield in future runs of the manufacturing process?

Solutions

First, let's split the `ChemicalManufacturingProcess` data set into the predictors and the response as follows:

```
> predictors <- subset(ChemicalManufacturingProcess,select= -Yield)
> yield <- subset(ChemicalManufacturingProcess,select="Yield")
```

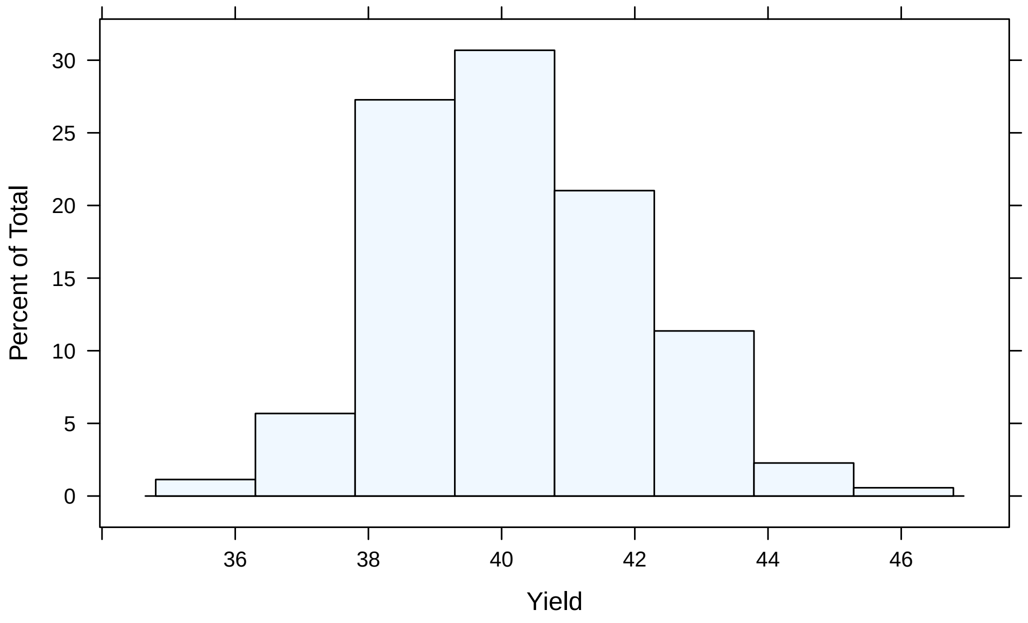

Figure 10: The response distribution for the chemical manufacturing data.

Then let's understand some fundamental characteristics of the response and predictors before modeling. The distribution of the response is presented in Figure 10; it is fairly symmetric and does not need transformation prior to analysis.

Are there any univariate relationships with the response? Because we have a small number of predictors, we can explore these relationships visually with a correlation plot as seen in Figure 11. The figure shows that many of the biological material predictors are highly positively correlated, and some of the manufacturing process predictors are positively and negatively correlated. Biological materials 2, 3, 6, and 8, and manufacturing processes 9 and 32 are highly positively correlated with Yield, whereas manufacturing processes 13 and 36 are highly negatively correlated with Yield.

A small percentage of the predictor set cells are missing. Specifically, 106 cells are missing out of 10032 or 1.06 percent. When we encounter missing cells, we also should investigate if any particular predictors or samples have a higher frequency of missingness. The top 10 missing predictors are:

```
                       Frequency
ManufacturingProcess03        15
ManufacturingProcess11        10
ManufacturingProcess10         9
ManufacturingProcess25         5
ManufacturingProcess26         5
ManufacturingProcess27         5
ManufacturingProcess28         5
```

```
ManufacturingProcess29         5
ManufacturingProcess30         5
ManufacturingProcess31         5
```

And the top 10 missing samples are:

```
    Frequency
1          16
172        11
173        11
174        11
175        11
176        11
23          4
2           3
3           3
4           3
```

It may be worthwhile to investigate why rows (samples) 1 and 172-176 have the highest frequency of missingness. Since no rows or columns have too much missing data, we can impute this information without perturbing the true underlying relationship between the predictors and the response. Let's first split the data into training (70%) and test (30%) sets. On that data, we will identify the appropriate Box-Cox transformation, centering and scaling parameters, and will impute using the k-nearest neighbor method. We will then apply those transformations and imputation to the test data.

```
> #Split data into training and test sets
> set.seed(517)
> trainingRows <- createDataPartition(yield$Yield,
+                                      p = 0.7,
+                                      list = FALSE)
>
> trainPredictors <- predictors[trainingRows,]
> trainYield <- yield[trainingRows,]
>
> testPredictors <- predictors[-trainingRows,]
> testYield <- yield[-trainingRows,]
>
> #Pre-process trainPredictors and apply to trainPredictors and testPredictors
> pp <- preProcess(trainPredictors,method=c("BoxCox","center","scale","knnImpute"))
> ppTrainPredictors <- predict(pp,trainPredictors)
> ppTestPredictors <- predict(pp,testPredictors)
```

Next, we should remove near-zero variance and highly correlated predictors.

```
> #Identify and remove NZV
> nzvpp = nearZeroVar(ppTrainPredictors)
> ppTrainPredictors <- ppTrainPredictors[-nzvpp]
> ppTestPredictors <- ppTestPredictors[-nzvpp]
>
> #Identify and remove highly correlated predictors
> predcorr = cor(ppTrainPredictors)
> highCorrpp <- findCorrelation(predcorr)
> ppTrainPredictors <- ppTrainPredictors[, -highCorrpp]
> ppTestPredictors <- ppTestPredictors[, -highCorrpp]
```

After pre-processing, 47 predictors remain for modeling.

Given the moderate pairwise correlations that were apparent in Figure 11, a dimension reduction or shrinkage technique would be an appropriate model for this data. Here we will tune a PLS model on the training data using 25 iterations of bootstrap cross-validation.

```
> set.seed(517)
> ctrl <- trainControl(method = "boot", number = 25)
>
> plsTune <- train(x = ppTrainPredictors, y = trainYield,
+                  method = "pls",
+                  tuneLength = 15,
+                  trControl = ctrl)
```

Figure 12 indicates that the optimal number of latent variables that maximizes R^2 is 3.

The relationship between the PLS components and the response can be seen in Figure 13, where the straight line is the linear regression fit, and the curved line is the loess fit. For the training data, the first two components are linearly related to yield with the first component having the stronger relationship with the response. The third component has a weak relationship with the response and may not be linear as indicated by the loess fit.

Next we will predict the response for the test set and compare the R^2 value with the one obtained through bootstrap cross-validation. The relationship is displayed in Figure 14, where the R^2 value is 0.592. Bootstrap cross-validation estimated that a three-component PLS model would have an R^2 value of 0.574.

Next, let's examine the variable importance values for the top 15 predictors for this data and model (Figure 15). For this data, the manufacturing process predictors dominate the top part of the list. This may be helpful for improving yield, since many of the manufacturing predictors can be controlled.

Finally, let's explore the relationships between the three top important predictors and the response. Figure 16 provides the univariate relationships with each of these predictors (after transformation) and the response. Clearly Process 9 and Process 32 have a positive relationship with Yield, while Process 13 has a negative relationship. If these manufacturing processes can be controlled, then altering these steps in the process to have higher (or lower) values could improve the overall Yield of the process. A statistically designed experiment could be used to investigate a causal relationship between the settings of these processes and the overall yield.

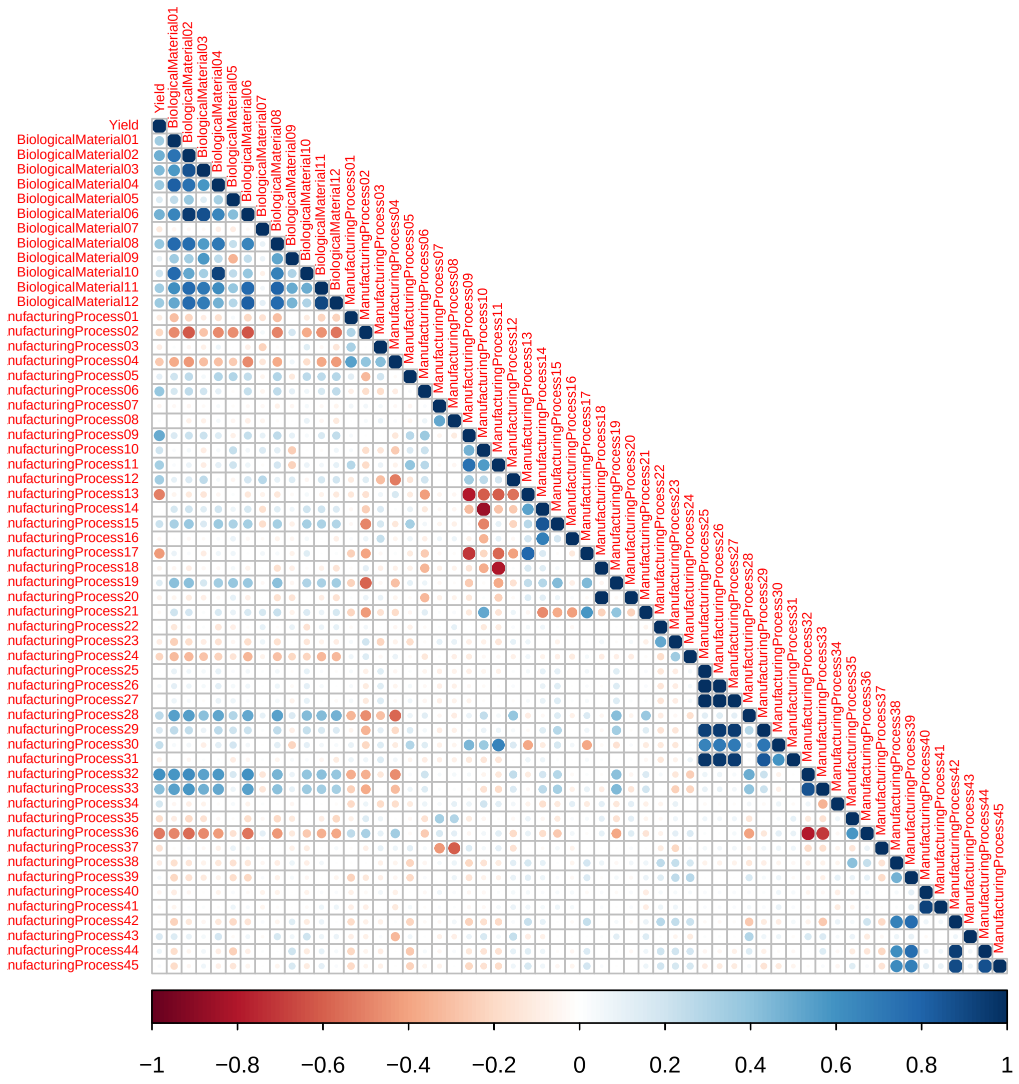

Figure 11: Scatterplot correlation matrix for the chemical manufacturing data.

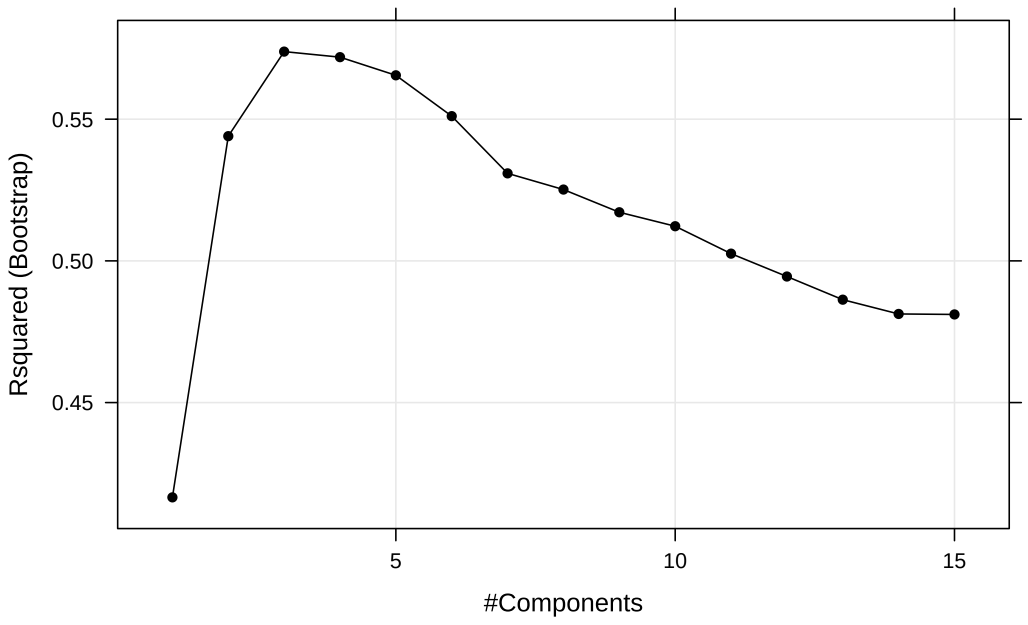

Figure 12: PLS tuning parameter profile for the manufacturing data

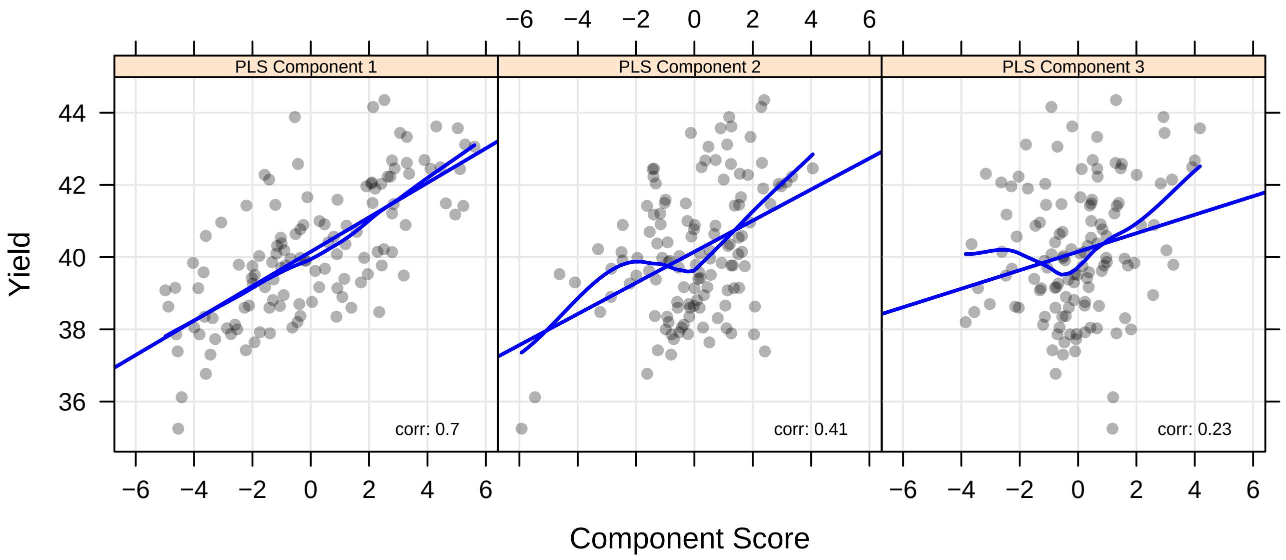

Figure 13: PLS scores versus response for the manufacturing data.

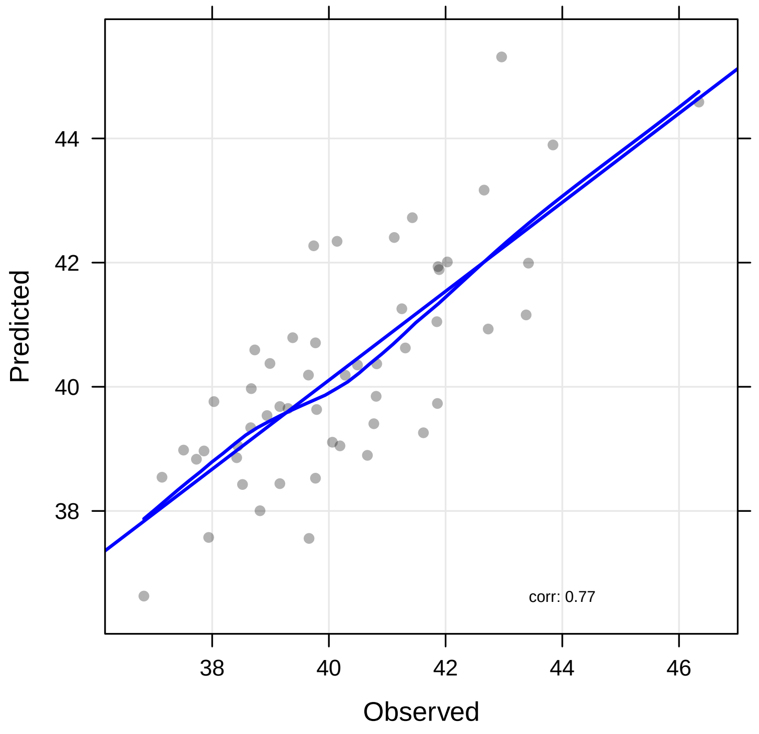

Figure 14: PLS predictions for the test set for the manufacturing data.

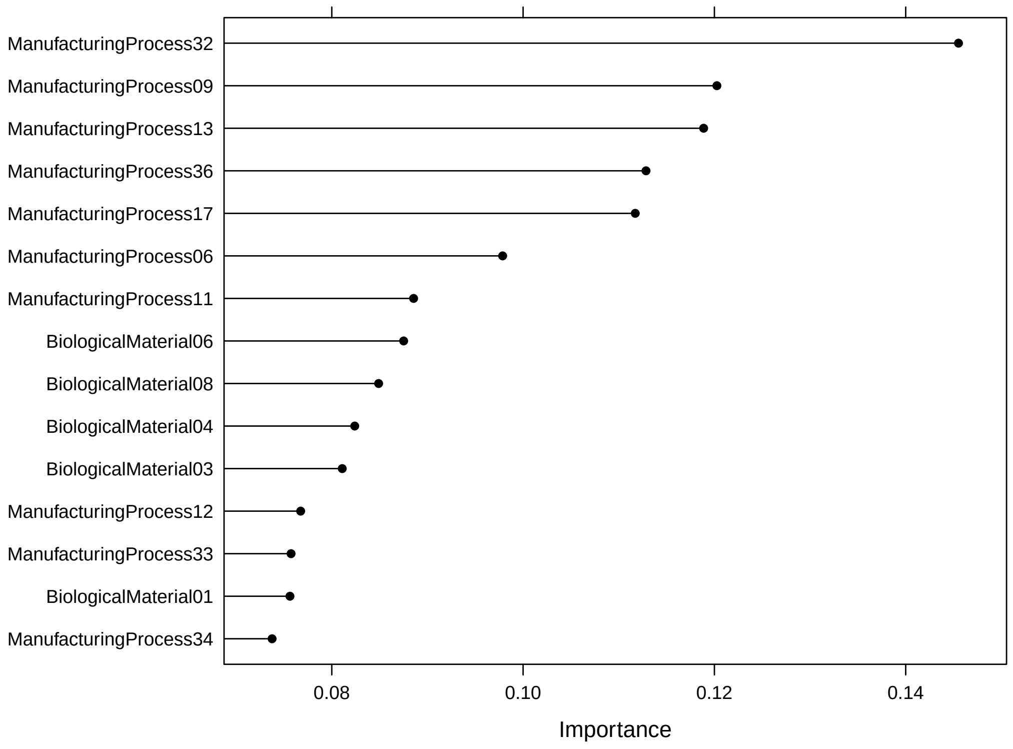

Figure 15: PLS variable importance scores for the manufacturing data.

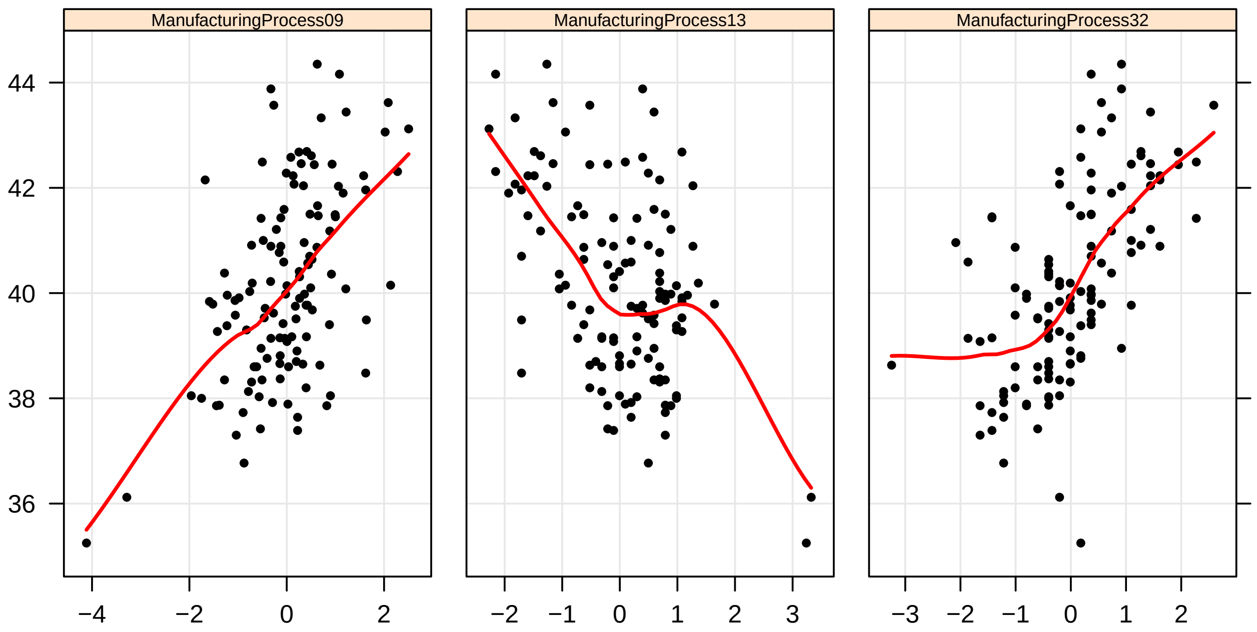

Figure 16: Scatter plots of the top 3 correlated predictors in the manufacturing data set after pre-processing.

Session Info

- R Under development (unstable) (2014-11-05 r66942), `x86_64-apple-darwin10.8.0`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: AppliedPredictiveModeling 1.1-6, caret 6.0-37, corrplot 0.73, doMC 1.3.3, earth 3.2-7, elasticnet 1.1, foreach 1.4.2, ggplot2 0.9.3.1, iterators 1.0.7, kernlab 0.9-19, knitr 1.6, lars 1.2, lattice 0.20-29, latticeExtra 0.6-26, mlbench 2.1-1, nnet 7.3-8, plotmo 1.3-3, plotrix 3.5-7, pls 2.4-3, RColorBrewer 1.0-5, reshape2 1.4, xtable 1.7-3
- Loaded via a namespace (and not attached): BradleyTerry2 1.0-5, brglm 0.5-9, car 2.0-21, class 7.3-11, cluster 1.15.3, codetools 0.2-9, colorspace 1.2-4, compiler 3.2.0, CORElearn 0.9.43, digest 0.6.4, e1071 1.6-3, evaluate 0.5.5, formatR 0.10, grid 3.2.0, gtable 0.1.2, gtools 3.4.1, highr 0.3, lme4 1.1-7, MASS 7.3-35, Matrix 1.1-4, minqa 1.2.3, munsell 0.4.2, nlme 3.1-118, nloptr 1.0.4, plyr 1.8.1, proto 0.3-10, RANN 2.4.1, Rcpp 0.11.2, rpart 4.1-8, scales 0.2.4, splines 3.2.0, stringr 0.6.2, tools 3.2.0