Regression

Lev Mazaev

Exercise 1 (Applied Predictive Modeling, p. 137)

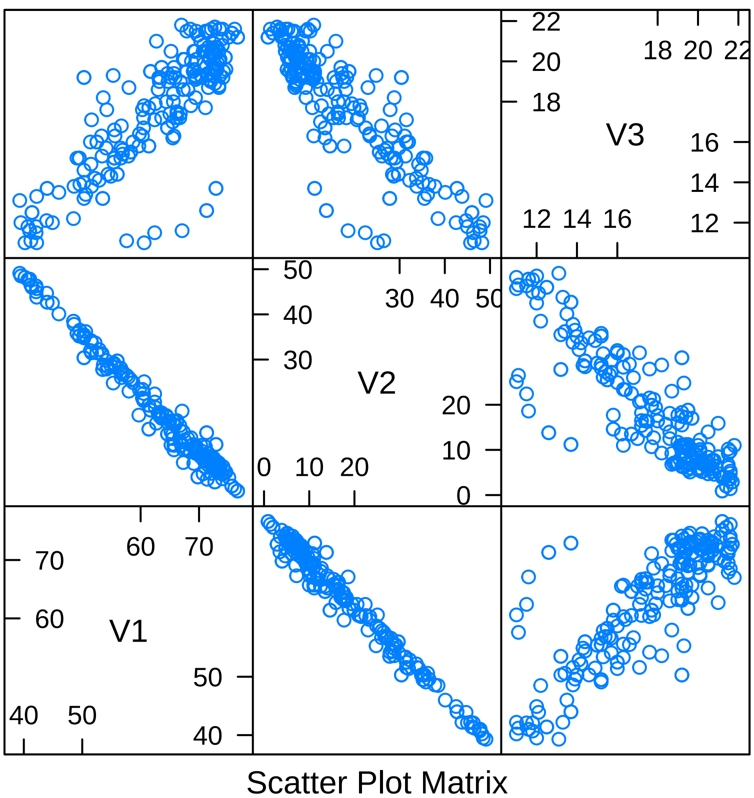
Answers to the questions are in the end of the document.

Loading packages and data

```
library(caret)
library(corrplot)
library(doMC)
registerDoMC(8)
data(tecator)
```

Checking the data

```
splom(~endpoints)
```




```
any(is.na(absorp))
```

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

```r
any(is.na(endpoints))
```

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

```r
colnames(absorp) <- paste0("V", 1:100)
head(absorp[, 1:6])
```

```
##           V1      V2      V3      V4      V5      V6
## [1,] 2.61776 2.61814 2.61859 2.61912 2.61981 2.62071
## [2,] 2.83454 2.83871 2.84283 2.84705 2.85138 2.85587
## [3,] 2.58284 2.58458 2.58629 2.58808 2.58996 2.59192
## [4,] 2.82286 2.82460 2.82630 2.82814 2.83001 2.83192
## [5,] 2.78813 2.78989 2.79167 2.79350 2.79538 2.79746
## [6,] 3.00993 3.01540 3.02086 3.02634 3.03190 3.03756
```

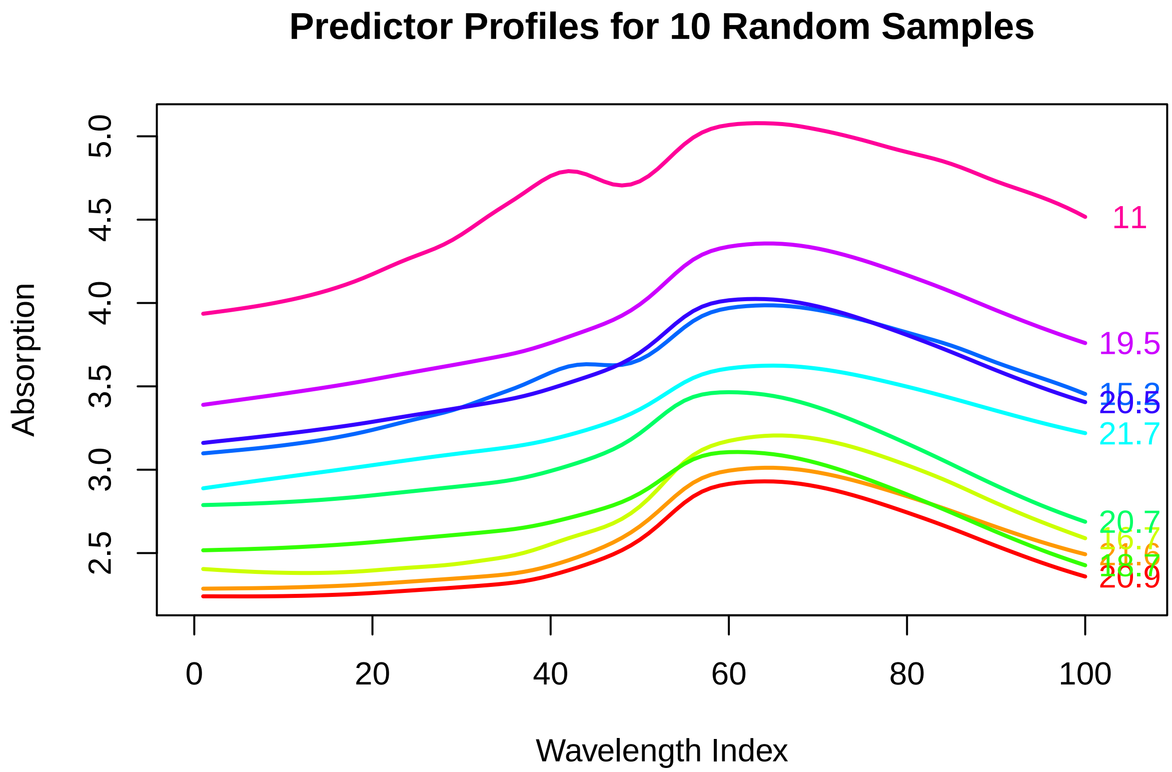
```r
head(endpoints)
```

```
##      [,1] [,2] [,3]
## [1,] 60.5 22.5 16.7
## [2,] 46.0 40.1 13.5
## [3,] 71.0  8.4 20.5
## [4,] 72.8  5.9 20.7
## [5,] 58.3 25.5 15.5
## [6,] 44.0 42.7 13.7
```

Plotting 10 random spectra

```r
p10randspectra <- function() {
set.seed(1)
inSubset <- sample(1:dim(endpoints)[1], 10)

absorpSubset <- absorp[inSubset,]
endpointSubset <- endpoints[inSubset, 3]

newOrder <- order(absorpSubset[,1])
absorpSubset <- absorpSubset[newOrder,]
endpointSubset <- endpointSubset[newOrder]

plotColors <- rainbow(10)

plot(absorpSubset[1,],
     type = "n",
     ylim = range(absorpSubset),
     xlim = c(0, 105),
     xlab = "Wavelength Index",
     ylab = "Absorption")

for(i in 1:10)
{
  points(absorpSubset[i,], type = "l", col = plotColors[i], lwd = 2)
  text(105, absorpSubset[i,100], endpointSubset[i], col = plotColors[i])
}
title("Predictor Profiles for 10 Random Samples")
```

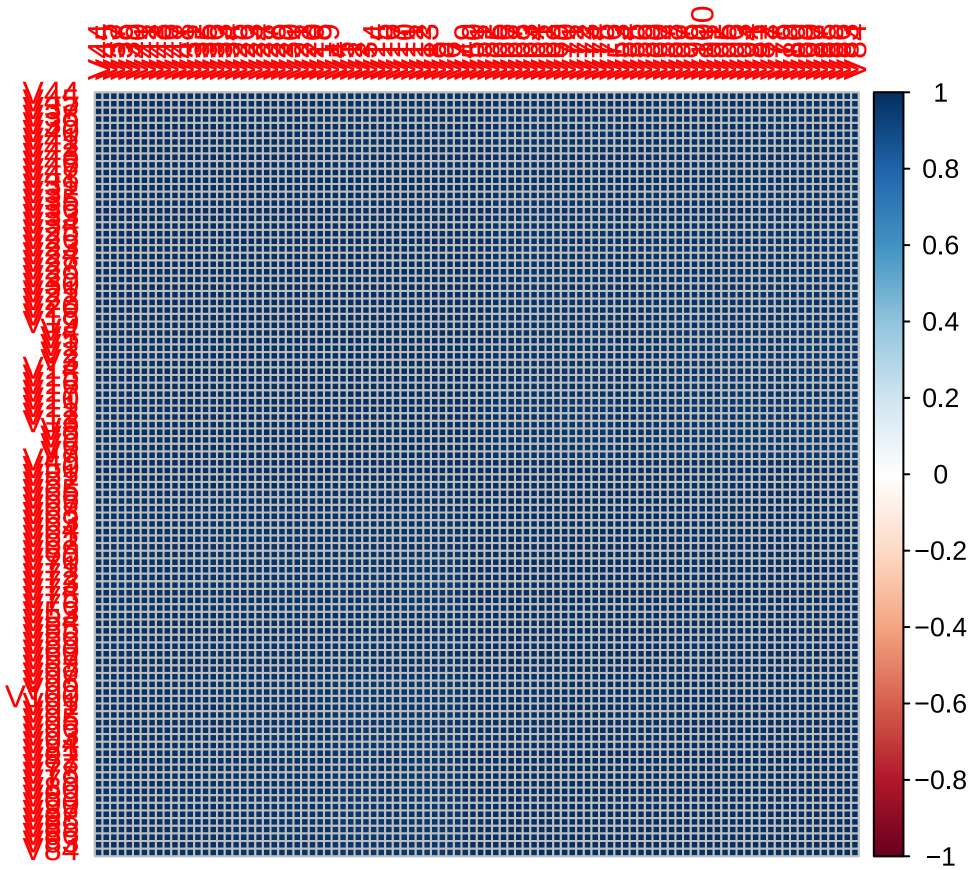
```r
}
p10randspectra()
```



Correlation matrix

All predictors are very highly correlated

```r
corrplot(corr = cor(absorp), order = "hclust")
```



Principal Component Analysis

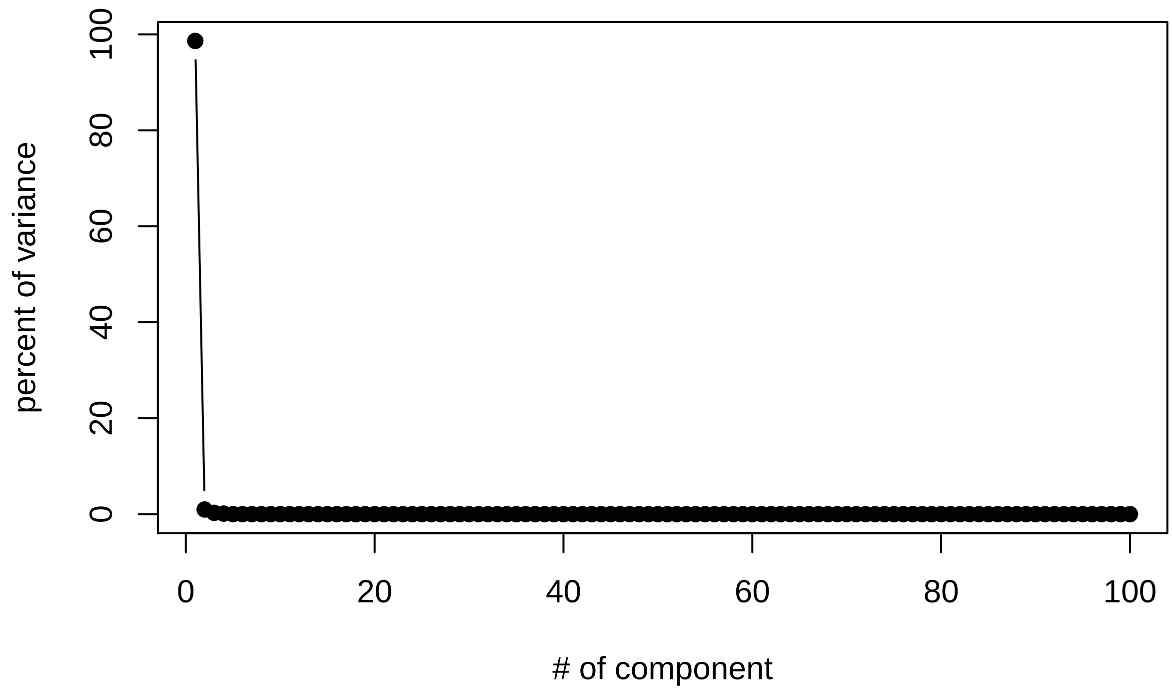
```r
pcaObject <- prcomp(absorp, center = TRUE, scale. = TRUE)
summary(pcaObject)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     9.9311 0.9847 0.52851 0.33827 0.08038 0.05123
## Proportion of Variance 0.9863 0.0097 0.00279 0.00114 0.00006 0.00003
## Cumulative Proportion  0.9863 0.9960 0.99875 0.99990 0.99996 0.99999
##                            PC7     PC8      PC9     PC10     PC11     PC12
## Standard deviation     0.02681 0.01961 0.008564 0.006739 0.004442 0.003361
## Proportion of Variance 0.00001 0.00000 0.000000 0.000000 0.000000 0.000000
## Cumulative Proportion  0.99999 1.00000 1.000000 1.000000 1.000000 1.000000
##                            PC13     PC14      PC15      PC16      PC17
## Standard deviation     0.001867 0.001377 0.0009449 0.0008641 0.0007558
## Proportion of Variance 0.000000 0.000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  1.000000 1.000000 1.0000000 1.0000000 1.0000000
##                             PC18      PC19      PC20      PC21      PC22
## Standard deviation     0.0006977 0.0005884 0.0004628 0.0003897 0.0003341
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##                             PC23      PC24      PC25     PC26      PC27
## Standard deviation     0.0003123 0.0002721 0.0002616 0.000211 0.0001954
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.000000 0.0000000
```

```
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.000000 1.0000000
##                             PC28      PC29      PC30      PC31      PC32
## Standard deviation     0.0001857 0.0001729 0.0001656 0.0001539 0.0001473
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##                             PC33      PC34      PC35      PC36     PC37
## Standard deviation     0.0001392 0.0001339 0.0001269 0.0001082 0.000104
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.0000000 1.000000
##                            PC38      PC39      PC40      PC41      PC42
## Standard deviation     9.98e-05 9.081e-05 8.668e-05 8.026e-05 7.762e-05
## Proportion of Variance 0.00e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.00e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC43      PC44      PC45     PC46      PC47
## Standard deviation     7.36e-05 6.808e-05 6.541e-05 6.44e-05 5.897e-05
## Proportion of Variance 0.00e+00 0.000e+00 0.000e+00 0.00e+00 0.000e+00
## Cumulative Proportion  1.00e+00 1.000e+00 1.000e+00 1.00e+00 1.000e+00
##                             PC48      PC49      PC50      PC51      PC52
## Standard deviation     5.422e-05 5.027e-05 4.893e-05 4.608e-05 4.419e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                             PC53      PC54    PC55     PC56      PC57
## Standard deviation     4.037e-05 3.854e-05 3.8e-05 3.64e-05 3.497e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.0e+00 0.00e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.0e+00 1.00e+00 1.000e+00
##                             PC58      PC59      PC60     PC61      PC62
## Standard deviation     3.443e-05 3.264e-05 3.104e-05 3.04e-05 2.959e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.00e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.00e+00 1.000e+00
##                             PC63      PC64      PC65      PC66      PC67
## Standard deviation     2.844e-05 2.699e-05 2.586e-05 2.388e-05 2.364e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                             PC68      PC69      PC70      PC71     PC72
## Standard deviation     2.284e-05 2.173e-05 2.058e-05 1.997e-05 1.93e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.00e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.00e+00
##                             PC73      PC74      PC75      PC76      PC77
## Standard deviation     1.854e-05 1.807e-05 1.728e-05 1.693e-05 1.612e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                             PC78      PC79      PC80      PC81      PC82
## Standard deviation     1.569e-05 1.516e-05 1.445e-05 1.408e-05 1.356e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                             PC83      PC84      PC85     PC86      PC87
## Standard deviation     1.275e-05 1.224e-05 1.178e-05 1.09e-05 1.045e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.00e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.00e+00 1.000e+00
##                             PC88      PC89      PC90     PC91      PC92
## Standard deviation     1.009e-05 9.396e-06 8.728e-06 8.27e-06 7.613e-06
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.00e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.00e+00 1.000e+00
##                            PC93      PC94      PC95      PC96      PC97
```

```
## Standard deviation     6.83e-06 6.383e-06 5.946e-06 5.478e-06 4.826e-06
## Proportion of Variance 0.00e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.00e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                             PC98      PC99     PC100
## Standard deviation     4.521e-06 4.164e-06 4.122e-06
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00
```

```r
plot(1:100, 100*summary(pcaObject)$importance[2, ], type = "b",
     pch = 19, xlab = "# of component", ylab = "percent of variance")
```




```r
rm(pcaObject)
```

The effective dimension of the data is 1 because PC1 catches 98.6% of variance

```r
PCAFit <- preProcess(absorp, method = c("pca"))
PCAFit
```

```
## Created from 215 samples and 100 variables
##
## Pre-processing:
##   - centered (100)
##   - ignored (0)
##   - principal component signal extraction (100)
##   - scaled (100)
##
## PCA needed 2 components to capture 95 percent of the variance
```

```r
rm(PCAFit)
```

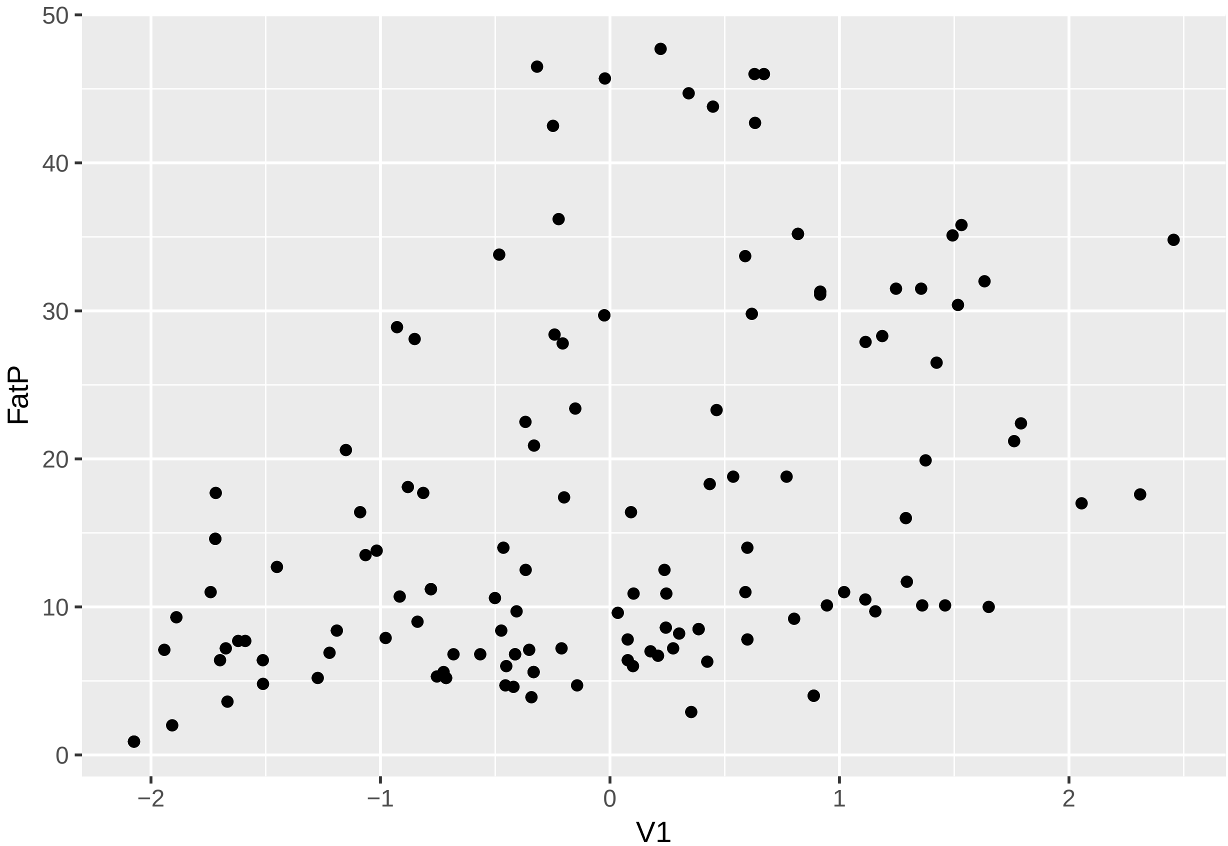
PCA from preProcess chooses 2 components

Pre-processing and splitting the data

```
trainrows <- createDataPartition(1:215, p = 0.6, list = TRUE)[[1]]
transFit <- preProcess(absorp, method = c("BoxCox", "center", "scale"))
transAbsorp <- as.data.frame(predict(transFit, absorp))
trainX <- transAbsorp[trainrows, ]
testX <- transAbsorp[-trainrows, ]
transEndpoints <- as.data.frame(endpoints)
trainY <- transEndpoints[trainrows, 2] # col #2 because only fat will be predicted
testY <- transEndpoints[-trainrows, 2]
trainingData <- cbind(trainX, trainY)
colnames(trainingData)[101] <- "FatP"
```

Visualization

```
ggplot(trainingData, aes(x = V1, y = FatP)) +
  geom_point()
```



Function to evaluate models (by defaultSummary)

```
modeval <- function(model, X = testX, Y = testY) {
  pred <- predict(model, X)
  values <- data.frame(obs = Y, pred = pred)
```

```r
  return(defaultSummary(values))
}
```

Ordinary Linear Regression, lm function

```r
lmbasic <- lm(FatP ~ ., data = trainingData)
summary(lmbasic)
```

```
##
## Call:
## lm(formula = FatP ~ ., data = trainingData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91560 -0.24840 -0.05153  0.26542  1.29998
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.828e+01  1.497e-01 122.056  < 2e-16 ***
## V1          -4.629e+02  1.548e+03  -0.299 0.766928
## V2          -1.942e+03  3.073e+03  -0.632 0.532207
## V3           1.014e+02  5.743e+03   0.018 0.986027
## V4           6.734e+03  9.207e+03   0.731 0.470211
## V5          -7.771e+03  1.084e+04  -0.717 0.478914
## V6           1.000e+04  9.226e+03   1.084 0.286833
## V7          -8.322e+03  5.288e+03  -1.574 0.126002
## V8           2.490e+03  3.673e+03   0.678 0.502943
## V9          -2.486e+02  3.045e+03  -0.082 0.935476
## V10         -2.583e+03  3.483e+03  -0.742 0.464092
## V11          8.111e+03  4.803e+03   1.689 0.101614
## V12         -1.025e+04  6.400e+03  -1.602 0.119595
## V13          1.307e+02  4.651e+03   0.028 0.977777
## V14         -8.119e+02  5.851e+03  -0.139 0.890573
## V15          7.559e+03  8.899e+03   0.849 0.402392
## V16         -4.688e+03  6.067e+03  -0.773 0.445755
## V17          4.404e+03  3.445e+03   1.278 0.210936
## V18          3.271e+02  3.928e+03   0.083 0.934185
## V19          7.619e+01  4.748e+03   0.016 0.987304
## V20         -3.289e+02  5.512e+03  -0.060 0.952818
## V21         -1.817e+04  6.598e+03  -2.753 0.009916 **
## V22          3.715e+04  9.633e+03   3.857 0.000565 ***
## V23         -3.531e+04  9.469e+03  -3.729 0.000799 ***
## V24          1.313e+04  6.903e+03   1.903 0.066708 .
## V25          1.519e+03  5.302e+03   0.287 0.776406
## V26          3.830e+03  4.064e+03   0.942 0.353512
## V27         -4.221e+03  4.224e+03  -0.999 0.325666
## V28         -3.479e+03  4.836e+03  -0.719 0.477428
## V29          7.589e+03  5.117e+03   1.483 0.148438
## V30         -6.984e+03  5.955e+03  -1.173 0.250090
## V31          4.768e+03  1.063e+04   0.448 0.657036
## V32         -1.622e+03  1.253e+04  -0.129 0.897879
## V33         -1.596e+01  9.312e+03  -0.002 0.998644
```

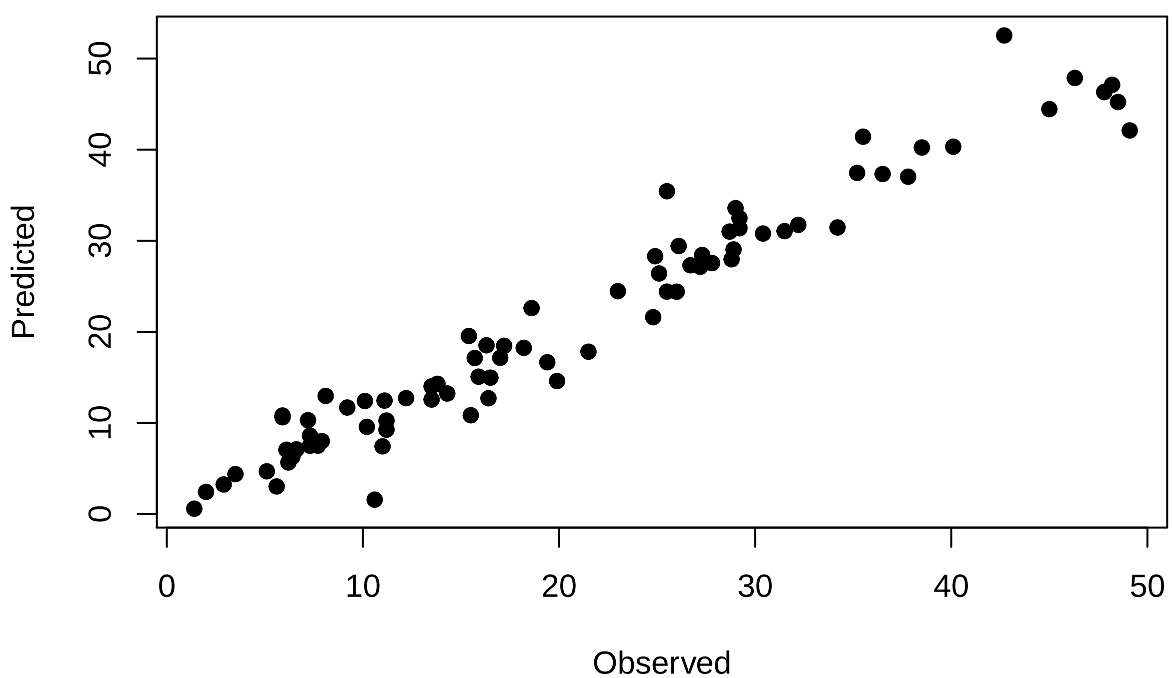
```
## V34         -4.273e+03  6.644e+03  -0.643 0.524968
## V35          3.266e+03  4.934e+03   0.662 0.513081
## V36         -3.711e+03  3.978e+03  -0.933 0.358339
## V37          2.642e+03  4.952e+03   0.533 0.597662
## V38          5.443e+03  6.558e+03   0.830 0.413123
## V39         -5.527e+03  8.271e+03  -0.668 0.509107
## V40          5.475e+01  1.056e+04   0.005 0.995898
## V41          2.243e+03  1.105e+04   0.203 0.840475
## V42          3.979e+03  9.709e+03   0.410 0.684874
## V43         -4.950e+03  5.932e+03  -0.834 0.410653
## V44         -5.449e+03  2.581e+03  -2.111 0.043211 *
## V45          1.088e+04  5.688e+03   1.913 0.065349 .
## V46         -7.409e+03  3.891e+03  -1.904 0.066501 .
## V47          3.031e+03  3.494e+03   0.868 0.392518
## V48         -1.012e+03  3.440e+03  -0.294 0.770722
## V49         -2.981e+03  4.001e+03  -0.745 0.462069
## V50          4.557e+03  5.327e+03   0.855 0.399107
## V51         -1.681e+02  5.243e+03  -0.032 0.974628
## V52          6.316e+02  4.197e+03   0.151 0.881376
## V53         -6.221e+03  3.551e+03  -1.752 0.090046 .
## V54          2.175e+03  3.860e+03   0.564 0.577210
## V55          9.994e+03  6.833e+03   1.463 0.153984
## V56         -1.185e+04  6.635e+03  -1.786 0.084257 .
## V57          2.176e+03  4.272e+03   0.509 0.614152
## V58          2.910e+03  3.049e+03   0.955 0.347458
## V59         -1.883e+03  2.393e+03  -0.787 0.437647
## V60          1.373e+03  2.123e+03   0.647 0.522782
## V61         -2.817e+03  2.888e+03  -0.975 0.337269
## V62          7.674e+03  3.709e+03   2.069 0.047282 *
## V63         -4.868e+03  4.459e+03  -1.092 0.283642
## V64         -4.672e+03  6.178e+03  -0.756 0.455434
## V65          7.054e+03  7.344e+03   0.960 0.344500
## V66         -2.219e+03  7.632e+03  -0.291 0.773231
## V67         -8.189e+03  8.457e+03  -0.968 0.340651
## V68          1.320e+04  8.196e+03   1.611 0.117665
## V69         -5.371e+03  6.807e+03  -0.789 0.436280
## V70         -3.230e+03  6.138e+03  -0.526 0.602614
## V71          5.738e+02  5.247e+03   0.109 0.913660
## V72          4.002e+03  4.147e+03   0.965 0.342322
## V73          3.961e+02  4.059e+03   0.098 0.922907
## V74         -5.646e+03  3.939e+03  -1.433 0.162080
## V75          6.331e+03  3.569e+03   1.774 0.086256 .
## V76         -1.783e+03  2.659e+03  -0.671 0.507642
## V77         -1.472e+03  3.114e+03  -0.473 0.639844
## V78          5.584e+03  4.082e+03   1.368 0.181494
## V79         -2.909e+03  4.033e+03  -0.721 0.476233
## V80         -4.971e+03  4.552e+03  -1.092 0.283530
## V81          7.322e+02  6.194e+03   0.118 0.906677
## V82         -2.500e+03  6.461e+03  -0.387 0.701508
## V83          9.551e+03  6.330e+03   1.509 0.141824
## V84         -2.057e+04  1.086e+04  -1.894 0.067901 .
## V85          2.190e+04  1.203e+04   1.820 0.078754 .
## V86         -2.656e+03  9.613e+03  -0.276 0.784211
## V87         -8.226e+03  8.442e+03  -0.974 0.337682
```

```
## V88          6.071e+03  7.268e+03   0.835 0.410133
## V89          3.563e+03  7.654e+03   0.466 0.644921
## V90         -9.866e+03  7.737e+03  -1.275 0.212038
## V91          7.244e+03  8.206e+03   0.883 0.384398
## V92          1.234e+03  9.512e+03   0.130 0.897651
## V93         -7.118e+03  9.387e+03  -0.758 0.454197
## V94          8.136e+03  9.444e+03   0.861 0.395809
## V95         -4.353e+03  5.011e+03  -0.869 0.391898
## V96         -1.561e+03  3.782e+03  -0.413 0.682691
## V97          1.169e+03  5.407e+03   0.216 0.830269
## V98         -2.322e+03  4.942e+03  -0.470 0.641867
## V99          9.215e+02  5.592e+03   0.165 0.870223
## V100         1.345e+03  2.353e+03   0.572 0.571628
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.847 on 30 degrees of freedom
## Multiple R-squared:  0.9989, Adjusted R-squared:  0.9952
## F-statistic: 270.8 on 100 and 30 DF,  p-value: < 2.2e-16
```

```
modeval(lmbasic)
```

```
##      RMSE  Rsquared       MAE
## 2.9617071 0.9519516 2.0495377
```

```
modstats <- t(as.data.frame(modeval(lmbasic)))
rownames(modstats)[1] <- "LM basic R"
plot(testY, predict(lmbasic, testX), pch = 19, xlab = "Observed", ylab = "Predicted")
```



Ordinary Linear Regression, lm from caret

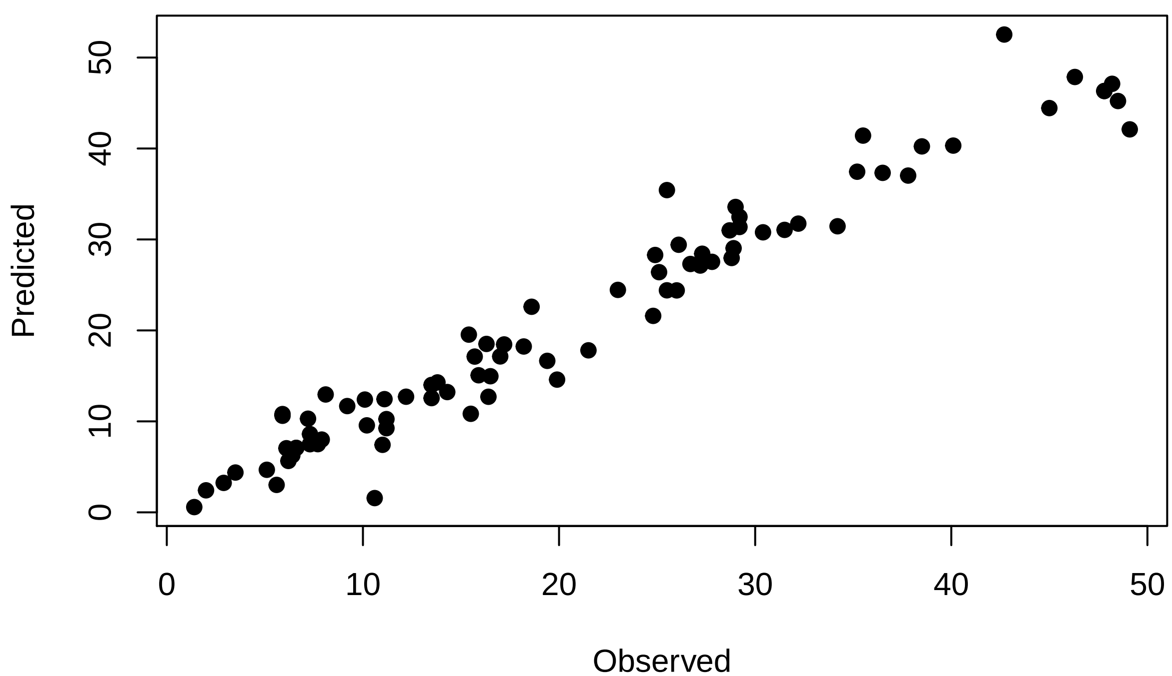
```r
ctrl <- trainControl(method = "cv", number = 10)
lmFit <- train(x = trainX, y = trainY, method = "lm", trControl = ctrl)
lmFit
```

```
## Linear Regression
##
## 131 samples
## 100 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 118, 118, 116, 118, 119, 118, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   4.893908  0.8749618  3.501878
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
modeval(lmFit)
```

```
##      RMSE  Rsquared       MAE
## 2.9617071 0.9519516 2.0495377
```

```r
modstats <- rbind(modstats, modeval(lmFit))
rownames(modstats)[2] <- "LM caret"
plot(testY, predict(lmFit, testX), pch = 19, xlab = "Observed", ylab = "Predicted")
```



Ordinary Linear Regression with filtering of highly correlated values

```r
tooHigh <- findCorrelation(cor(trainX), cutoff = 0.99)
length(tooHigh)
```

```
## [1] 98
```

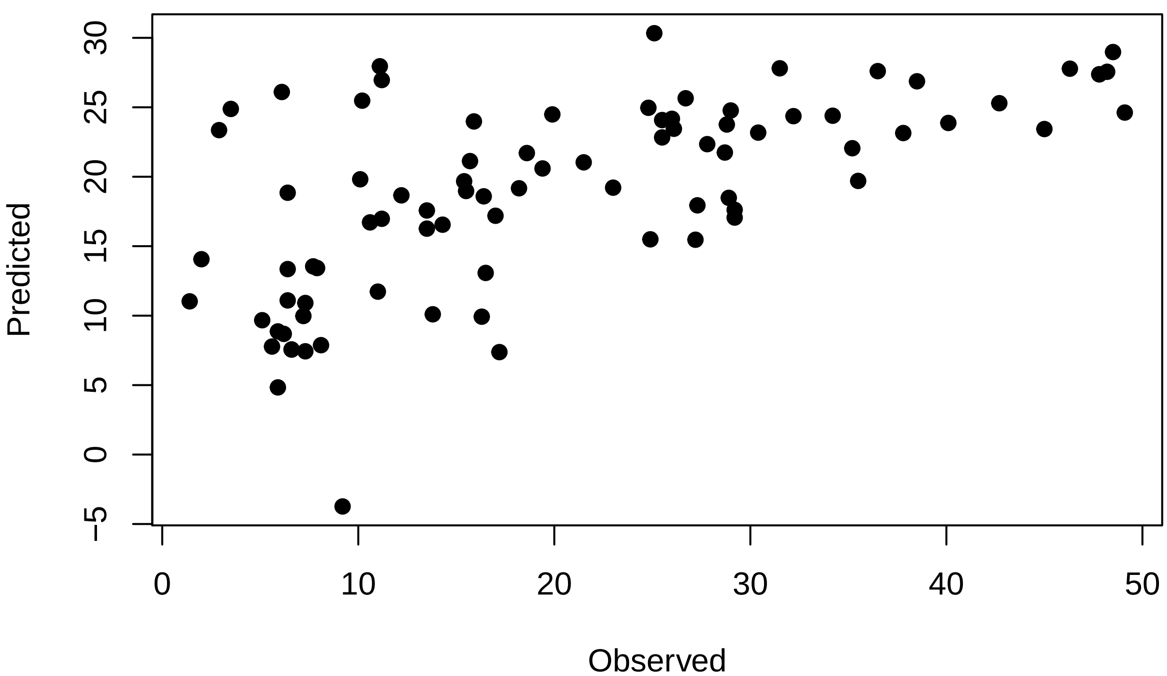
```r
trainXfiltered <- trainX[, -tooHigh]
testXfiltered <- testX[, -tooHigh]
lmFiltered <- train(x = trainXfiltered, y = trainY,
                    method = "lm", trControl = ctrl)
lmFiltered
```

```
## Linear Regression
##
## 131 samples
##   2 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 119, 119, 117, 118, 117, 118, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   9.557742  0.4272833  7.737826
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
modeval(lmFiltered, X = testXfiltered)
```

```
##        RMSE   Rsquared        MAE
## 10.3175997  0.4119963  8.0231636
```

```r
modstats <- rbind(modstats, modeval(lmFiltered, X = testXfiltered))
rownames(modstats)[3] <- "LM filtered"
plot(testY, predict(lmFiltered, testXfiltered), pch = 19, xlab = "Observed", ylab = "Predicted")
```



Robust Linear Regression, rlm from MASS

```r
library(MASS)
rlmFit <- rlm(FatP ~ ., data = trainingData)
```

```
## Warning in rlm.default(x, y, weights, method = method, wt.method =
## wt.method, : 'rlm' failed to converge in 20 steps
```

```r
summary(rlmFit)
```

```
##
## Call: rlm(formula = FatP ~ ., data = trainingData)
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.4619189 -0.0199005 -0.0003726  0.0181482  3.3265732
##
## Coefficients:
##             Value       Std. Error  t value
## (Intercept)     18.3761      0.0138  1335.0451
## V1            2770.6347    142.2602    19.4758
## V2           -7298.8485    282.5009   -25.8366
## V3            3063.5439    527.9737     5.8025
## V4            5566.3191    846.3872     6.5766
## V5          -10883.4683    996.3101   -10.9238
## V6           13669.5039    848.0823    16.1181
## V7           -9272.4878    486.0814   -19.0760
## V8            4231.2882    337.6115    12.5330
## V9             435.6881    279.9365     1.5564
## V10          -3262.2660    320.1872   -10.1886
## V11           4990.9074    441.4776    11.3050
## V12          -6964.9200    588.3377   -11.8383
## V13          -2548.5699    427.5765    -5.9605
## V14          -1570.7126    537.8888    -2.9201
## V15          13268.9496    818.0370    16.2205
## V16          -9029.0281    557.7159   -16.1893
## V17           5507.6272    316.7185    17.3897
## V18           2552.8091    361.0687     7.0701
## V19          -5270.2011    436.5035   -12.0737
## V20           3281.4370    506.7061     6.4760
## V21         -15316.1554    606.5358   -25.2519
## V22          26730.6917    885.4885    30.1875
## V23         -24734.8057    870.4901   -28.4148
## V24           8049.0797    634.5428    12.6848
## V25           4914.6899    487.3985    10.0835
## V26           2153.3341    373.5707     5.7642
## V27          -5686.2240    388.3103   -14.6435
## V28          -3822.6584    444.5125    -8.5997
## V29          10318.0984    470.3602    21.9366
## V30          -6896.4375    547.3866   -12.5988
## V31           4311.6326    977.2368     4.4121
## V32          -6855.7322   1152.0293    -5.9510
## V33           6733.4215    856.0403     7.8658
## V34          -6030.1009    610.7172    -9.8738
## V35            393.3382    453.5466     0.8672
```

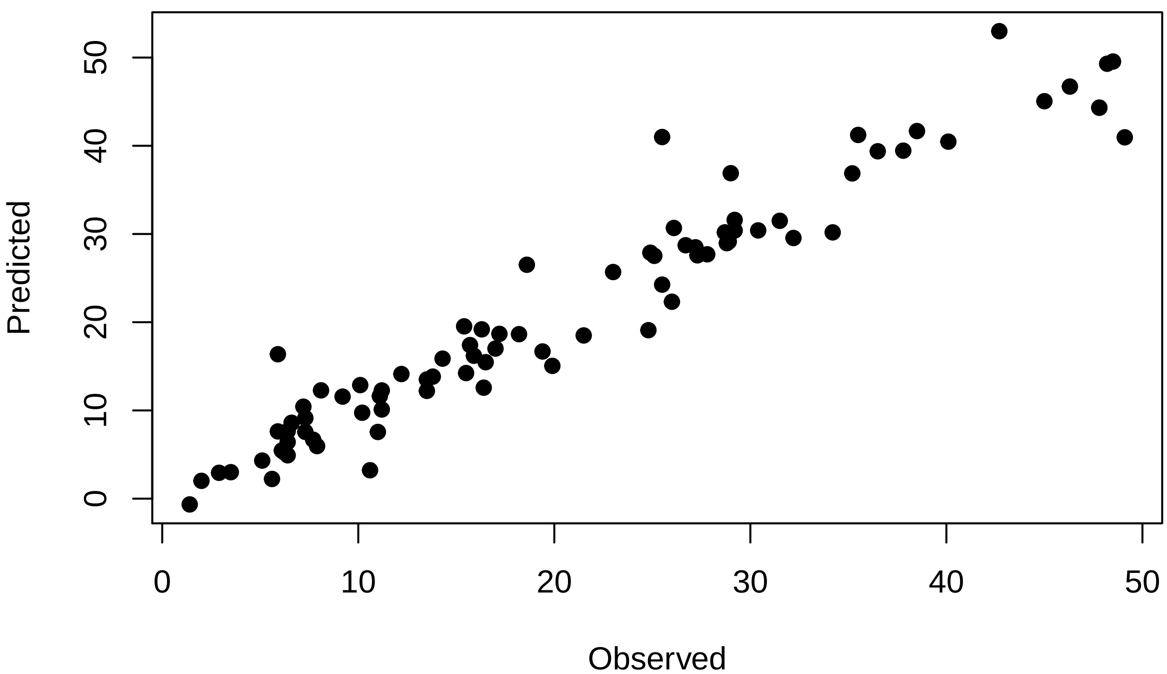
```
## V36           1064.8944    365.6967      2.9120
## V37            244.1951    455.2480      0.5364
## V38           3614.6406    602.8809      5.9956
## V39          -6752.2724    760.2826     -8.8813
## V40           7375.1906    970.8101      7.5969
## V41          -3230.7212   1015.6892     -3.1808
## V42           4344.0515    892.5309      4.8671
## V43          -3894.0512    545.3409     -7.1406
## V44          -6076.1117    237.2634    -25.6091
## V45          11252.3001    522.8719     21.5202
## V46          -8577.8163    357.6493    -23.9839
## V47           5207.7855    321.1987     16.2136
## V48          -1470.1216    316.2524     -4.6486
## V49          -3823.0969    367.7824    -10.3950
## V50           2521.4502    489.6783      5.1492
## V51           3537.4867    481.9563      7.3399
## V52           -987.2358    385.8046     -2.5589
## V53          -6137.7572    326.4689    -18.8004
## V54           3838.7904    354.8047     10.8194
## V55           5912.6381    628.1547      9.4127
## V56          -9554.4487    609.9312    -15.6648
## V57           4701.4954    392.6766     11.9729
## V58           -583.2631    280.3006     -2.0808
## V59           -926.8589    220.0126     -4.2128
## V60           2584.4644    195.1934     13.2405
## V61          -4769.0360    265.5223    -17.9610
## V62           4901.9000    340.9797     14.3759
## V63           3677.4417    409.9242      8.9710
## V64          -9375.2076    567.9233    -16.5079
## V65           4869.9198    675.1466      7.2131
## V66          -1837.3990    701.6072     -2.6188
## V67          -2566.4278    777.4626     -3.3010
## V68           8768.4660    753.4153     11.6383
## V69          -6659.1568    625.7213    -10.6424
## V70            -81.0452    564.2104     -0.1436
## V71           -377.6398    482.3735     -0.7829
## V72           4121.4926    381.2608     10.8102
## V73          -5490.7170    373.1020    -14.7164
## V74           1087.3778    362.0761      3.0032
## V75           5755.8616    328.1296     17.5414
## V76          -2967.4297    244.4496    -12.1392
## V77            168.9009    286.2800      0.5900
## V78           6493.9110    375.2388     17.3061
## V79          -6141.4021    370.6981    -16.5671
## V80          -5107.9144    418.4830    -12.2058
## V81           1953.1412    569.3530      3.4305
## V82          -2180.4742    593.9336     -3.6712
## V83           5140.4764    581.9129      8.8338
## V84         -12238.7120    998.1301    -12.2616
## V85          17668.0528   1105.9253     15.9758
## V86          -2842.9511    883.7185     -3.2170
## V87          -6406.6042    776.0650     -8.2552
## V88           2324.6917    668.0762      3.4797
## V89           6392.2113    703.6409      9.0845
```

```
## V90          -9756.6156    711.2069    -13.7184
## V91          -1402.1908    754.3505     -1.8588
## V92          17930.1291    874.3721     20.5063
## V93         -20289.8542    862.9119    -23.5132
## V94          13695.8545    868.1389     15.7761
## V95          -4763.8510    460.6303    -10.3420
## V96          -4584.4534    347.6886    -13.1855
## V97           6688.3539    497.0259     13.4568
## V98         -10357.9420    454.2568    -22.8020
## V99           7525.8916    514.0890     14.6393
## V100          -643.7613    216.2599     -2.9768
##
## Residual standard error: 0.03033 on 30 degrees of freedom
```

```r
modeval(rlmFit)
```

```
##      RMSE  Rsquared       MAE
## 3.6285964 0.9316943 2.4430944
```

```r
modstats <- rbind(modstats, modeval(rlmFit))
rownames(modstats)[4] <- "RLM MASS"
plot(testY, predict(rlmFit, testX), pch = 19, xlab = "Observed", ylab = "Predicted")
```



Robust Linear Regression with PCA

```r
rlmPCA <- train(x = trainX, y = trainY, method = "rlm", preProcess = "pca",
                trControl = ctrl)
summary(rlmPCA)
```

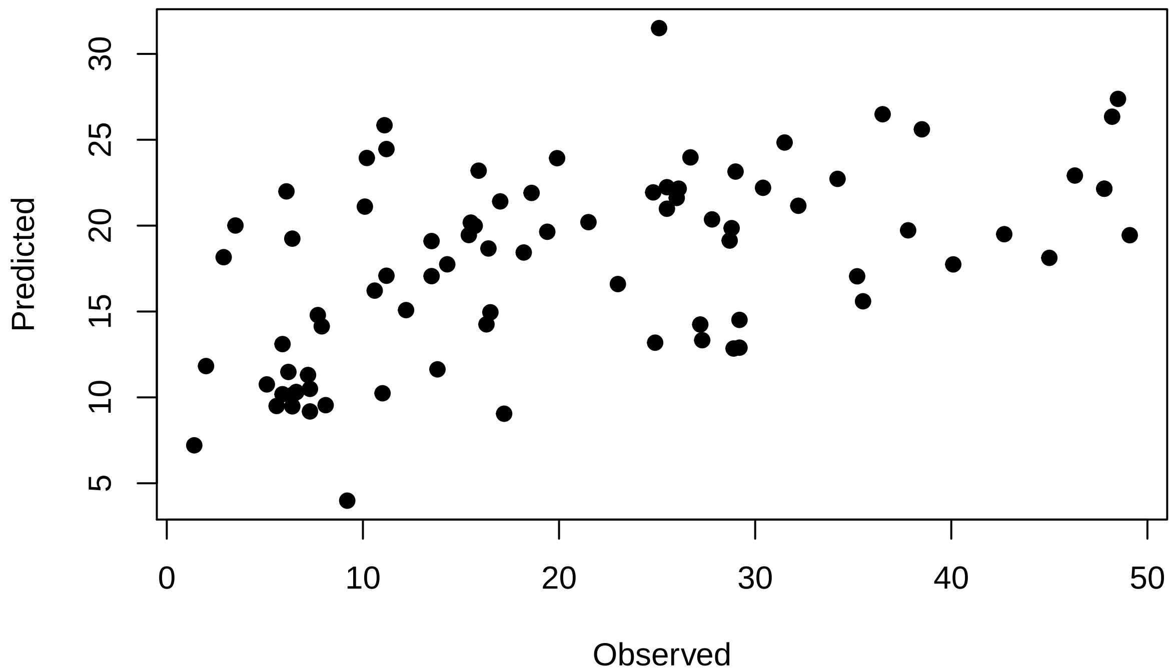
```
##
## Call: rlm(formula = .outcome ~ ., data = dat, psi = psi)
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -17.800  -6.627  -1.751   6.751  30.745
##
## Coefficients:
##             Value   Std. Error t value
## (Intercept) 16.1510  0.8912    18.1232
## PC1          0.5524  0.0901     6.1334
## PC2          3.0393  0.9011     3.3728
##
## Residual standard error: 9.871 on 128 degrees of freedom
```

```
modeval(rlmPCA)
```

```
##       RMSE   Rsquared        MAE
## 11.5720269  0.2833121  9.1018698
```

```
modstats <- rbind(modstats, modeval(rlmPCA))
rownames(modstats)[5] <- "RLM PCA"
plot(testY, predict(rlmPCA, testX), pch = 19, xlab = "Observed", ylab = "Predicted")
```



Partial Least Squares Regression

```
indx <- createFolds(trainY, returnTrain = TRUE)
ctrl <- trainControl(method = "cv", index = indx)
plsTune <- train(x = trainX, y = trainY, method = "pls",
                 tuneGrid = expand.grid(ncomp = 1:50),
                 trControl = ctrl)
```

Tuning parameters

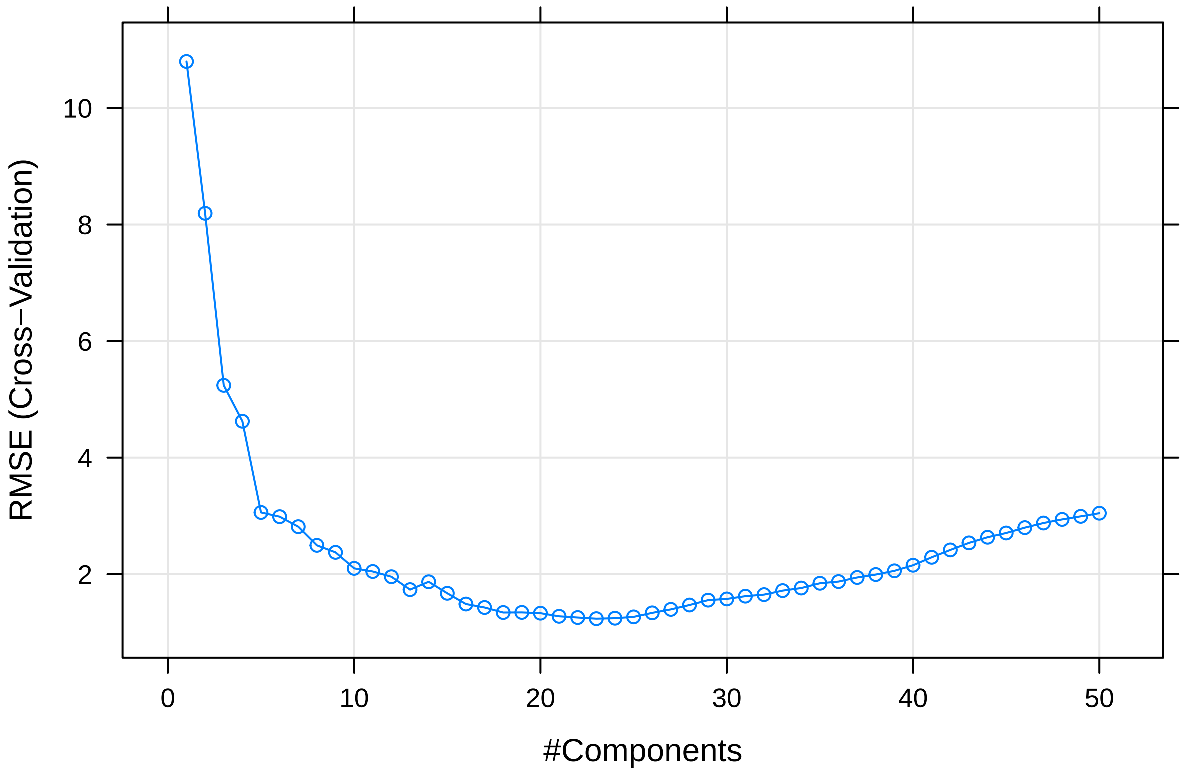
```
plsTune
```

```
## Partial Least Squares
##
```

```
## 131 samples
## 100 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 117, 117, 118, 118, 118, 118, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared   MAE
##    1     10.797966  0.2310967  8.7416942
##    2      8.192915  0.5648930  6.7464233
##    3      5.240568  0.8104156  4.1004251
##    4      4.624434  0.8642693  3.6080863
##    5      3.059443  0.9488733  2.2870680
##    6      2.986761  0.9541787  2.2186577
##    7      2.815218  0.9611686  2.0701101
##    8      2.495234  0.9710269  1.7576628
##    9      2.374636  0.9706905  1.7333109
##   10      2.099915  0.9763563  1.5480682
##   11      2.046205  0.9800377  1.4812418
##   12      1.956509  0.9816527  1.4097997
##   13      1.734992  0.9839708  1.3243399
##   14      1.869780  0.9812099  1.3514968
##   15      1.672174  0.9844472  1.2201141
##   16      1.487821  0.9866369  1.0984136
##   17      1.428482  0.9878811  1.0601455
##   18      1.342269  0.9887071  1.0152220
##   19      1.344269  0.9887766  1.0142185
##   20      1.330680  0.9890851  1.0110143
##   21      1.277281  0.9891429  0.9573668
##   22      1.256334  0.9896627  0.9510145
##   23      1.236562  0.9899599  0.9421029
##   24      1.244488  0.9900481  0.9461620
##   25      1.267352  0.9895641  0.9552685
##   26      1.336248  0.9884864  1.0056685
##   27      1.396058  0.9875957  1.0332871
##   28      1.470713  0.9860245  1.0756039
##   29      1.554574  0.9843602  1.1072995
##   30      1.575198  0.9840256  1.1301841
##   31      1.622658  0.9827225  1.1530157
##   32      1.649369  0.9821588  1.1810998
##   33      1.718049  0.9806263  1.2260404
##   34      1.762263  0.9794269  1.2455132
##   35      1.844402  0.9775971  1.2945704
##   36      1.873360  0.9770406  1.3258212
##   37      1.943670  0.9753431  1.3694065
##   38      1.993889  0.9737307  1.3899052
##   39      2.057341  0.9723603  1.4283037
##   40      2.153605  0.9700275  1.5060714
##   41      2.289832  0.9661782  1.6045963
##   42      2.416229  0.9619112  1.6876382
##   43      2.537148  0.9582574  1.7569513
##   44      2.634238  0.9550149  1.8044361
##   45      2.706725  0.9521107  1.8319188
```

```
##   46      2.798918  0.9480944  1.8584917
##   47      2.878299  0.9451495  1.8938869
##   48      2.940079  0.9425120  1.9341781
##   49      2.992495  0.9402794  1.9636487
##   50      3.046340  0.9382202  2.0011228
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 23.
```

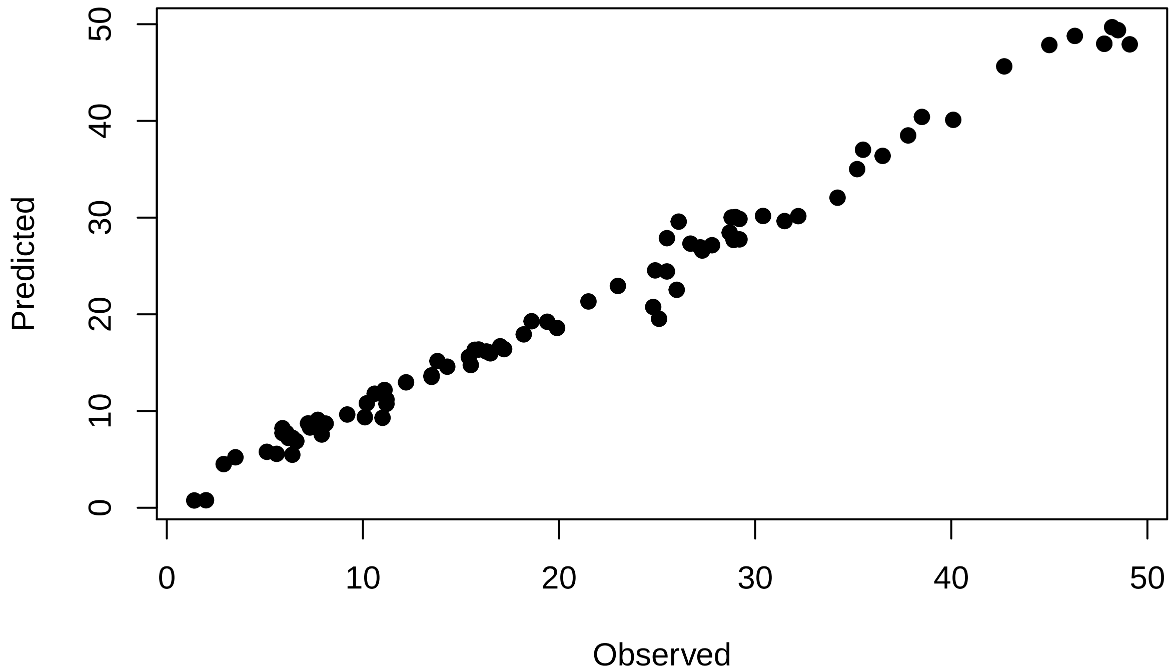
```
plot(plsTune)
```




```
modeval(plsTune)
```

```
##     RMSE Rsquared      MAE
## 1.450754 0.987997 1.059324
```

```
modstats <- rbind(modstats, modeval(plsTune))
rownames(modstats)[6] <- "PLSR"
plot(testY, predict(plsTune, testX), pch = 19, xlab = "Observed", ylab = "Predicted")
```



Principal Component Regression

```r
pcrTune <- train(x = trainX, y = trainY, method = "pcr",
                 tuneGrid = expand.grid(ncomp = 1:50),
                 trControl = ctrl)
```

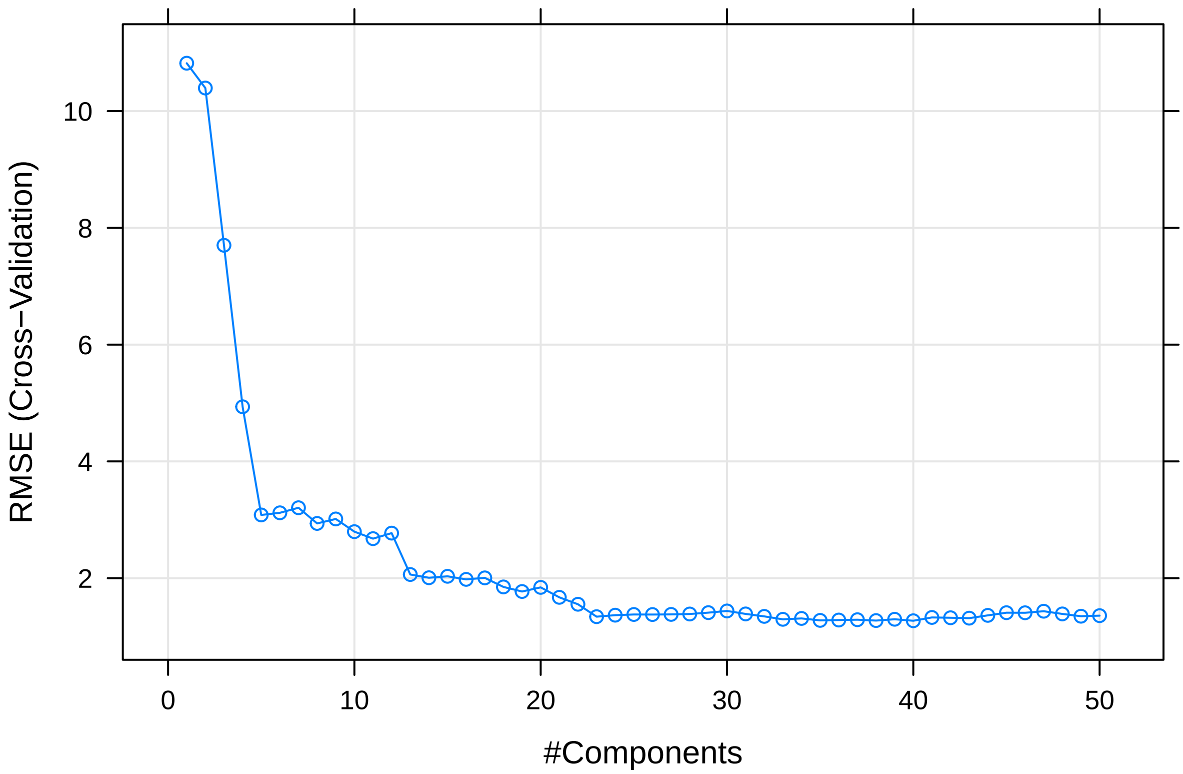
Tuning parameters

```r
pcrTune
```

```
## Principal Component Analysis
##
## 131 samples
## 100 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 117, 117, 118, 118, 118, 118, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared   MAE
##    1     10.820029  0.2279488  8.7661806
##    2     10.395319  0.2848395  8.2763549
##    3      7.701887  0.6016426  6.1666383
##    4      4.935712  0.8413751  3.8252846
##    5      3.082908  0.9478825  2.3047073
##    6      3.119444  0.9461726  2.3485515
##    7      3.206411  0.9442564  2.3917926
##    8      2.937604  0.9559360  2.1733786
##    9      3.014202  0.9550826  2.1952601
##   10      2.796865  0.9616962  2.0389795
```

```
##   11      2.676876  0.9644601  1.9022231
##   12      2.771657  0.9602406  1.9632980
##   13      2.062902  0.9799680  1.5039108
##   14      2.007329  0.9798668  1.4465465
##   15      2.032164  0.9798037  1.4509418
##   16      1.979465  0.9810269  1.4438343
##   17      2.005106  0.9807589  1.4493639
##   18      1.850263  0.9832113  1.3330298
##   19      1.771701  0.9822700  1.2734702
##   20      1.842635  0.9813318  1.3081510
##   21      1.671934  0.9836261  1.2024552
##   22      1.553051  0.9860597  1.1803341
##   23      1.340847  0.9887691  1.0171450
##   24      1.366147  0.9883475  1.0344072
##   25      1.378025  0.9881334  1.0403588
##   26      1.377411  0.9882482  1.0333659
##   27      1.379409  0.9882609  1.0366295
##   28      1.386430  0.9880385  1.0379671
##   29      1.409747  0.9874717  1.0635313
##   30      1.438352  0.9872350  1.0868188
##   31      1.387631  0.9878453  1.0789480
##   32      1.345721  0.9885926  1.0680052
##   33      1.295375  0.9891948  1.0293526
##   34      1.310491  0.9890261  1.0280200
##   35      1.277249  0.9896162  0.9789182
##   36      1.283031  0.9896241  0.9702687
##   37      1.288451  0.9893819  0.9548171
##   38      1.273877  0.9896686  0.9359205
##   39      1.296278  0.9893254  0.9612144
##   40      1.270711  0.9900889  0.9431444
##   41      1.327426  0.9892806  0.9761604
##   42      1.322145  0.9893100  0.9861686
##   43      1.315049  0.9893868  0.9790014
##   44      1.362870  0.9889922  1.0194829
##   45      1.408792  0.9883322  1.0435443
##   46      1.407356  0.9883655  1.0419257
##   47      1.434110  0.9876521  1.0659104
##   48      1.387823  0.9883715  1.0410124
##   49      1.348606  0.9885151  1.0109201
##   50      1.357893  0.9883561  1.0157296
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 40.
```

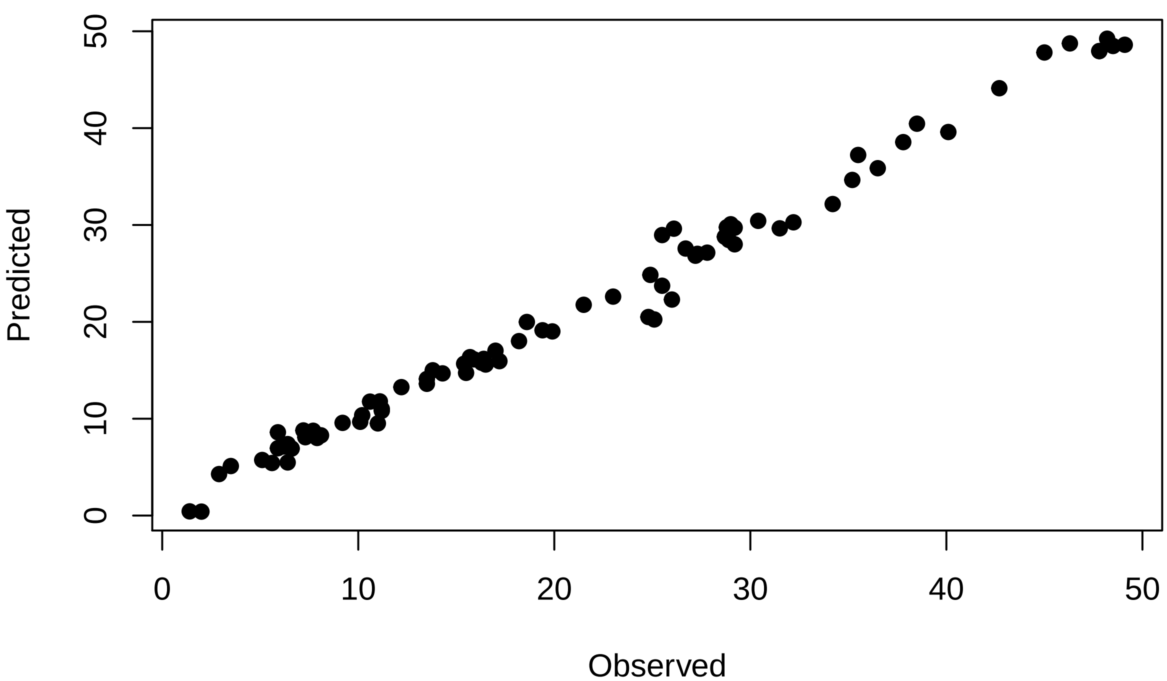
```r
plot(pcrTune)
```




```r
modeval(pcrTune)
```

```
##      RMSE  Rsquared       MAE
## 1.4114441 0.9885807 1.0174803
```

```r
modstats <- rbind(modstats, modeval(pcrTune))
rownames(modstats)[7] <- "PCR"
plot(testY, predict(pcrTune, testX), pch = 19, xlab = "Observed", ylab = "Predicted")
```



Ridge Regression

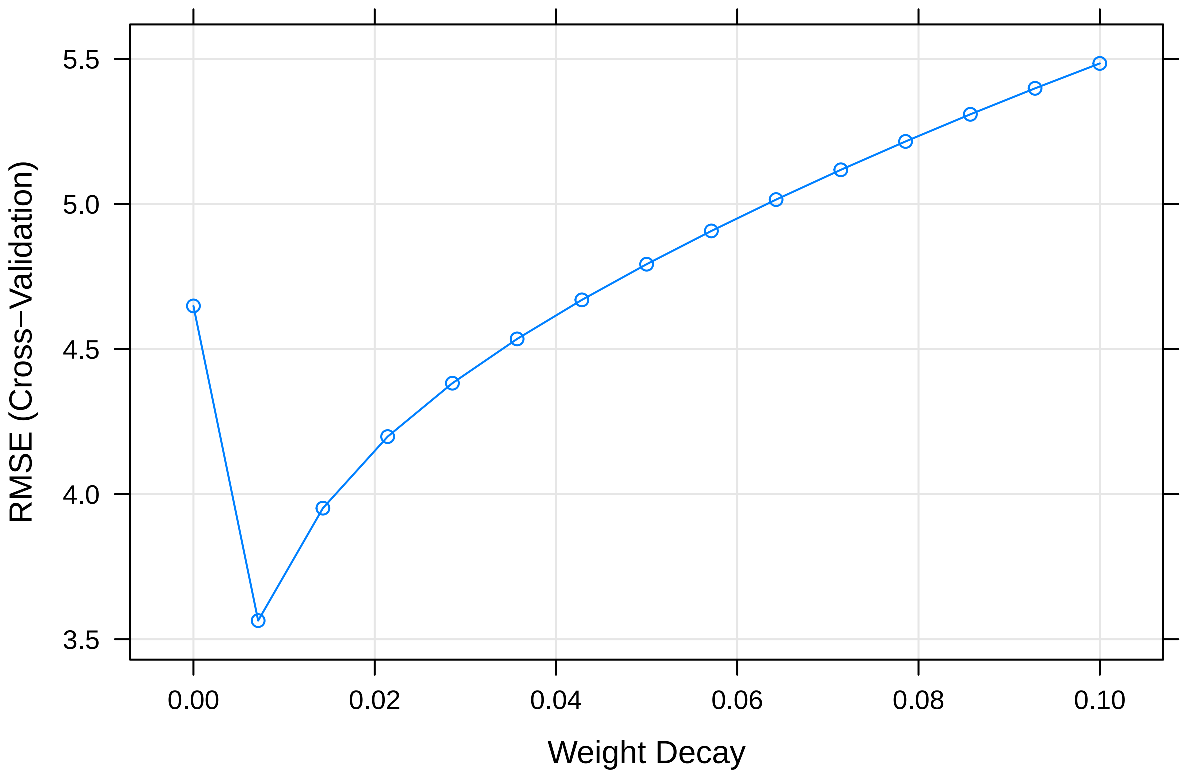
```
ridgeGrid <- data.frame(.lambda = seq(0, .1, length = 15))
ridgeRegFit <- train(x = trainX, y = trainY, method = "ridge", tuneGrid = ridgeGrid,
                     trControl = ctrl)
```

Tuning parameters

```
ridgeRegFit
```

```
## Ridge Regression
##
## 131 samples
## 100 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 117, 117, 118, 118, 118, 118, ...
## Resampling results across tuning parameters:
##
##   lambda       RMSE      Rsquared   MAE
##   0.000000000  4.648676  0.8739361  3.053590
##   0.007142857  3.564346  0.9264257  2.704078
##   0.014285714  3.951852  0.9052381  2.977535
##   0.021428571  4.198426  0.8910699  3.121035
##   0.028571429  4.382477  0.8805028  3.234836
##   0.035714286  4.535012  0.8718656  3.342973
##   0.042857143  4.669545  0.8643297  3.443875
##   0.050000000  4.792486  0.8574560  3.531894
##   0.057142857  4.907102  0.8509979  3.616112
##   0.064285714  5.015175  0.8448108  3.707965
##   0.071428571  5.117760  0.8388070  3.792516
##   0.078571429  5.215540  0.8329326  3.871539
##   0.085714286  5.308993  0.8271537  3.947053
##   0.092857143  5.398487  0.8214493  4.020327
##   0.100000000  5.484324  0.8158064  4.092073
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was lambda = 0.007142857.
```

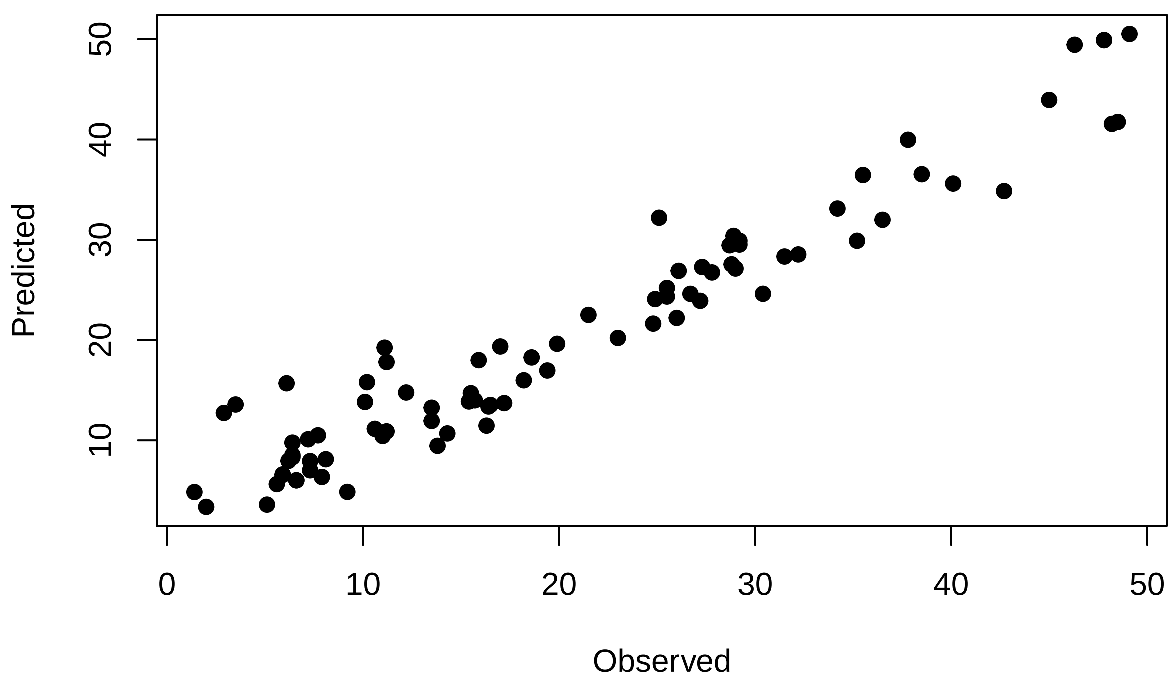
```
plot(ridgeRegFit)
```




```r
modeval(ridgeRegFit)
```

```
##      RMSE  Rsquared       MAE
## 3.5626739 0.9278507 2.6659641
```

```r
modstats <- rbind(modstats, modeval(ridgeRegFit))
rownames(modstats)[8] <- "Ridge"
plot(testY, predict(ridgeRegFit, testX), pch = 19, xlab = "Observed", ylab = "Predicted")
```



LASSO Regression

```r
enetGrid <- expand.grid(.lambda = c(0, 0.01, .1),
                        .fraction = seq(0.5, 1, length = 20))
enetTune <- train(x = trainX, y = trainY, method = "enet", tuneGrid = enetGrid,
                  trControl = ctrl)
```

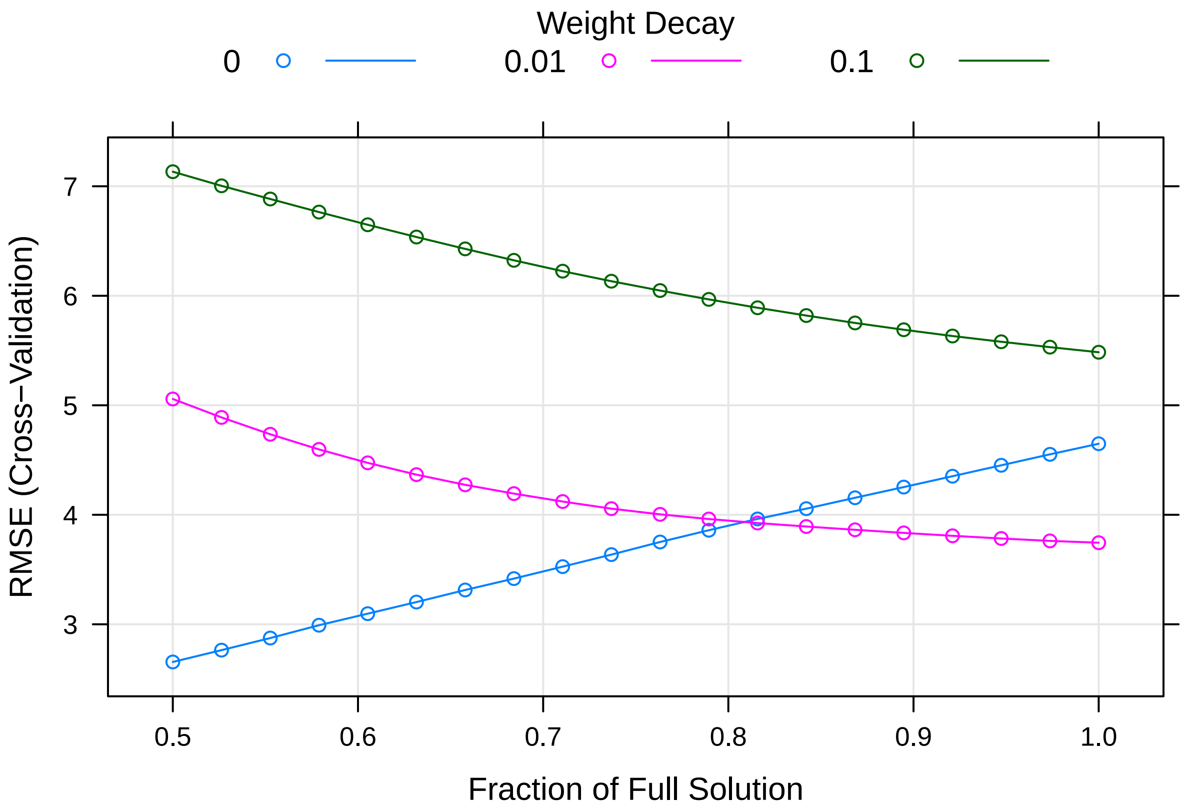
Tuning parameters

```r
enetTune
```

```
## Elasticnet
##
## 131 samples
## 100 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 117, 117, 118, 118, 118, 118, ...
## Resampling results across tuning parameters:
##
##   lambda  fraction   RMSE      Rsquared   MAE
##   0.00    0.5000000  2.655872  0.9553970  1.776683
##   0.00    0.5263158  2.764026  0.9517389  1.838204
##   0.00    0.5526316  2.874207  0.9479933  1.904889
##   0.00    0.5789474  2.991536  0.9439906  1.978107
##   0.00    0.6052632  3.097154  0.9404009  2.044172
##   0.00    0.6315789  3.203109  0.9366824  2.107306
##   0.00    0.6578947  3.313290  0.9327638  2.174620
##   0.00    0.6842105  3.417302  0.9286165  2.240152
##   0.00    0.7105263  3.526303  0.9241090  2.305848
##   0.00    0.7368421  3.636393  0.9197645  2.373436
##   0.00    0.7631579  3.751580  0.9151503  2.437882
##   0.00    0.7894737  3.859141  0.9104812  2.496985
##   0.00    0.8157895  3.962087  0.9059274  2.561530
##   0.00    0.8421053  4.055855  0.9016803  2.630589
##   0.00    0.8684211  4.155228  0.8971424  2.701036
##   0.00    0.8947368  4.253113  0.8925933  2.770857
##   0.00    0.9210526  4.352258  0.8879968  2.841411
##   0.00    0.9473684  4.451800  0.8833440  2.912634
##   0.00    0.9736842  4.552012  0.8785606  2.984549
##   0.00    1.0000000  4.648676  0.8739361  3.053590
##   0.01    0.5000000  5.057557  0.8596438  3.835138
##   0.01    0.5263158  4.889250  0.8672786  3.680157
##   0.01    0.5526316  4.735316  0.8739756  3.551369
##   0.01    0.5789474  4.596967  0.8797555  3.437292
##   0.01    0.6052632  4.474406  0.8848446  3.341581
##   0.01    0.6315789  4.366714  0.8892359  3.255599
##   0.01    0.6578947  4.273453  0.8930382  3.181319
##   0.01    0.6842105  4.193409  0.8962758  3.123712
##   0.01    0.7105263  4.120861  0.8993046  3.072212
##   0.01    0.7368421  4.055950  0.9020514  3.027957
##   0.01    0.7631579  4.004021  0.9042753  3.000367
##   0.01    0.7894737  3.961440  0.9061097  2.977436
```

```
##   0.01    0.8157895  3.924519  0.9077530  2.955062
##   0.01    0.8421053  3.892900  0.9092024  2.934966
##   0.01    0.8684211  3.863021  0.9106925  2.915994
##   0.01    0.8947368  3.834561  0.9121567  2.897791
##   0.01    0.9210526  3.808118  0.9135219  2.880316
##   0.01    0.9473684  3.783534  0.9147905  2.863686
##   0.01    0.9736842  3.761562  0.9159236  2.849526
##   0.01    1.0000000  3.745081  0.9167936  2.839046
##   0.10    0.5000000  7.133142  0.7042959  5.607195
##   0.10    0.5263158  7.004225  0.7158628  5.486939
##   0.10    0.5526316  6.883820  0.7262456  5.374492
##   0.10    0.5789474  6.764364  0.7361912  5.264182
##   0.10    0.6052632  6.648373  0.7454467  5.158882
##   0.10    0.6315789  6.536535  0.7539657  5.055459
##   0.10    0.6578947  6.428407  0.7618546  4.953551
##   0.10    0.6842105  6.323749  0.7691552  4.854308
##   0.10    0.7105263  6.224720  0.7757464  4.763113
##   0.10    0.7368421  6.133017  0.7815685  4.678516
##   0.10    0.7631579  6.047820  0.7867325  4.598173
##   0.10    0.7894737  5.967188  0.7914492  4.520387
##   0.10    0.8157895  5.890551  0.7957659  4.447580
##   0.10    0.8421053  5.819292  0.7996186  4.384847
##   0.10    0.8684211  5.752000  0.8031288  4.328816
##   0.10    0.8947368  5.689656  0.8062664  4.276132
##   0.10    0.9210526  5.632556  0.8090275  4.226434
##   0.10    0.9473684  5.580039  0.8114849  4.179379
##   0.10    0.9736842  5.530842  0.8137461  4.134220
##   0.10    1.0000000  5.484324  0.8158064  4.092073
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were fraction = 0.5 and lambda = 0.
```

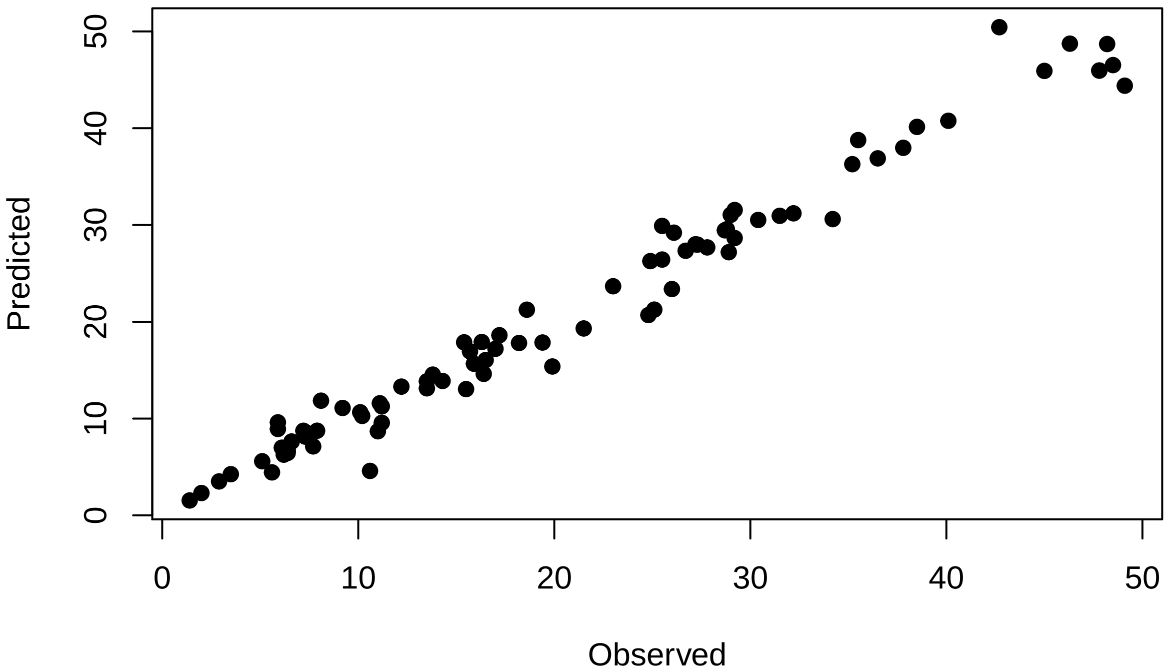
```r
plot(enetTune)
```




```r
modeval(enetTune)
```

```
##      RMSE  Rsquared       MAE
## 2.1013149 0.9748699 1.5205947
```

```r
modstats <- rbind(modstats, modeval(enetTune))
rownames(modstats)[9] <- "LASSO"
plot(testY, predict(enetTune, testX), pch = 19, xlab = "Observed", ylab = "Predicted")
```



Comparison of models

```
modstats
```

```
##                  RMSE  Rsquared      MAE
## LM basic R   2.961707 0.9519516 2.049538
## LM caret     2.961707 0.9519516 2.049538
## LM filtered 10.317600 0.4119963 8.023164
## RLM MASS     3.628596 0.9316943 2.443094
## RLM PCA     11.572027 0.2833121 9.101870
## PLSR         1.450754 0.9879970 1.059324
## PCR          1.411444 0.9885807 1.017480
## Ridge        3.562674 0.9278507 2.665964
## LASSO        2.101315 0.9748699 1.520595
```

Answers

b. The effective dimension of the data according to the PCA test is 1, because PC1 catches 98.6% of variance.

c. Evaluations and plots for tuning parameters of some models are aforementioned.

d. Principal Component Regression has the best predictive ability: the lowest $RMSE$ and the highest R^2. Some models, such as RLM with PCA and LM with filtering of highly correlated predictors, are significantly worse than others. Possibly these models loose some valuable information contained in predictors that were removed due to high correlation or during PCA.

e. I would use PCR or PLSR for predicting the fat content of a sample because they show the highest and very similar performance on the test data. LASSO may also be considered, all other models models are outperformed by these ones.