Establishing a predictive model to predict fat content in meat using Infared Absorbance

Abdelsalam A

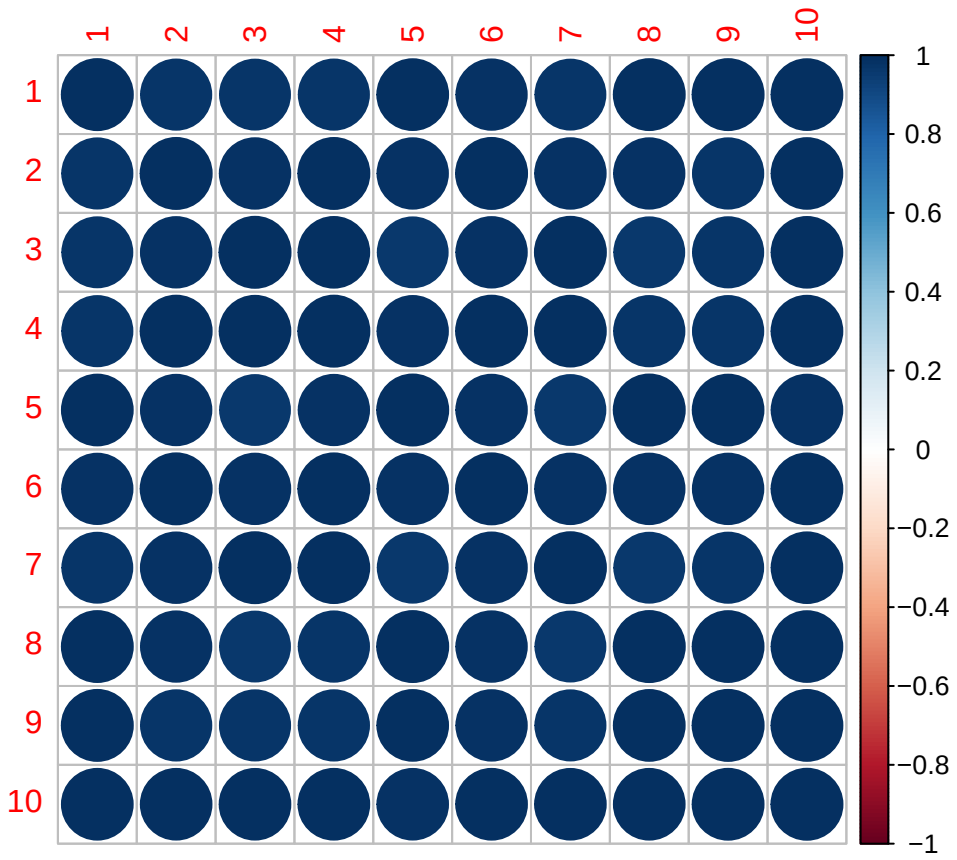
Traditional analytical chemistry methods are used to accurately measure the fat content in foods. However, these can be expensive and sometimes overkill in cases like routine quality control, educational settings, and other non-critical applications.

The *Theory of IR Spectroscopy* proposes that unique molecular structures absorb frequencies of light differently. We exploit this by establishing a predictive relationship between the IR spectrum and fat content using a *dataset of meat samples measured by Tecator*.

The dataset is explained further by the following permission note: "For each meat sample the data consists of a 100 channel spectrum of absorbances and the contents of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The three contents, measured in percent, are determined by analytic chemistry."

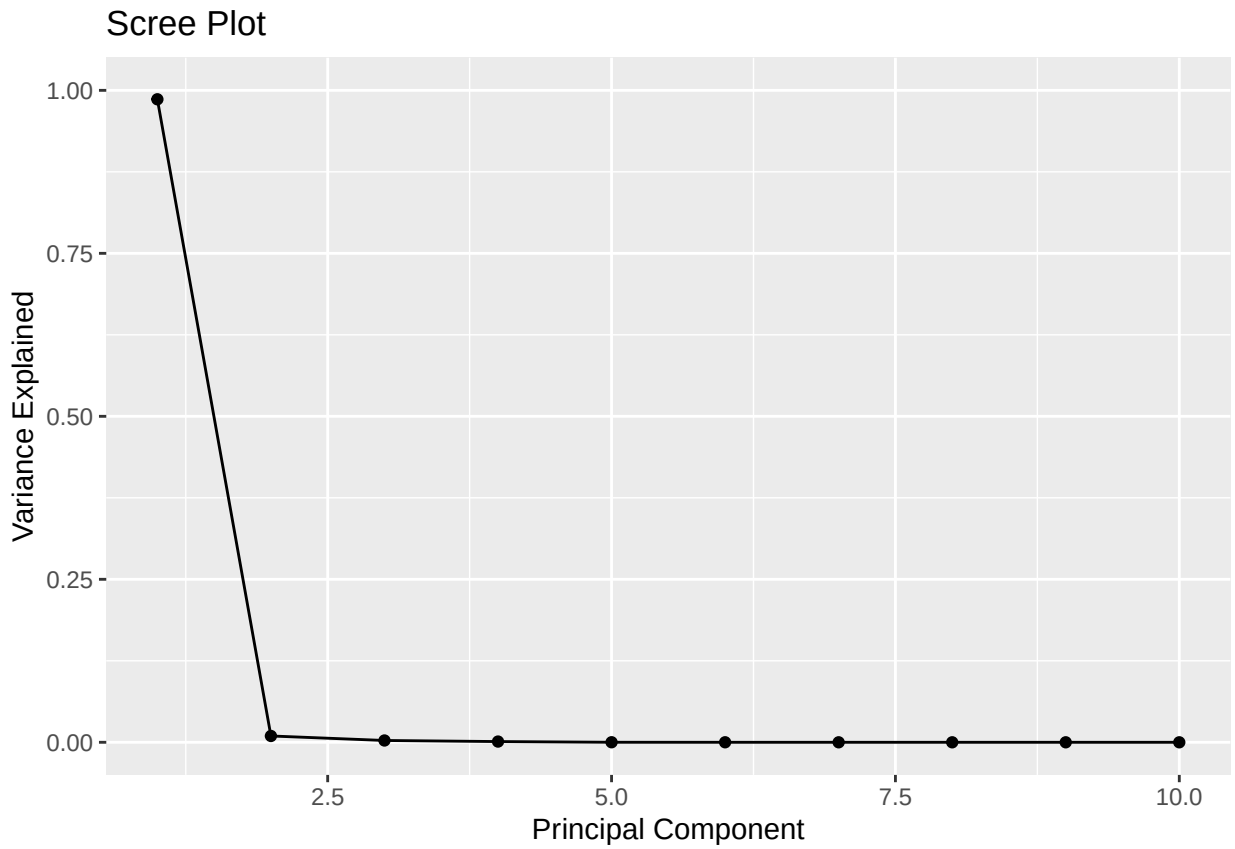
Since our objective is some LR model, we need to check for whether there's correlation between the predictors to see whether PCA is necessary for pre-processing:

```
# Take an random set of 10 absorbances and construct the correlation matrix
set.seed(1)
randP <- sample(1:dim(absorp)[2], 10)
absorpSubset <- absorp[,randP]
corrplot(cor(absorpSubset))
```



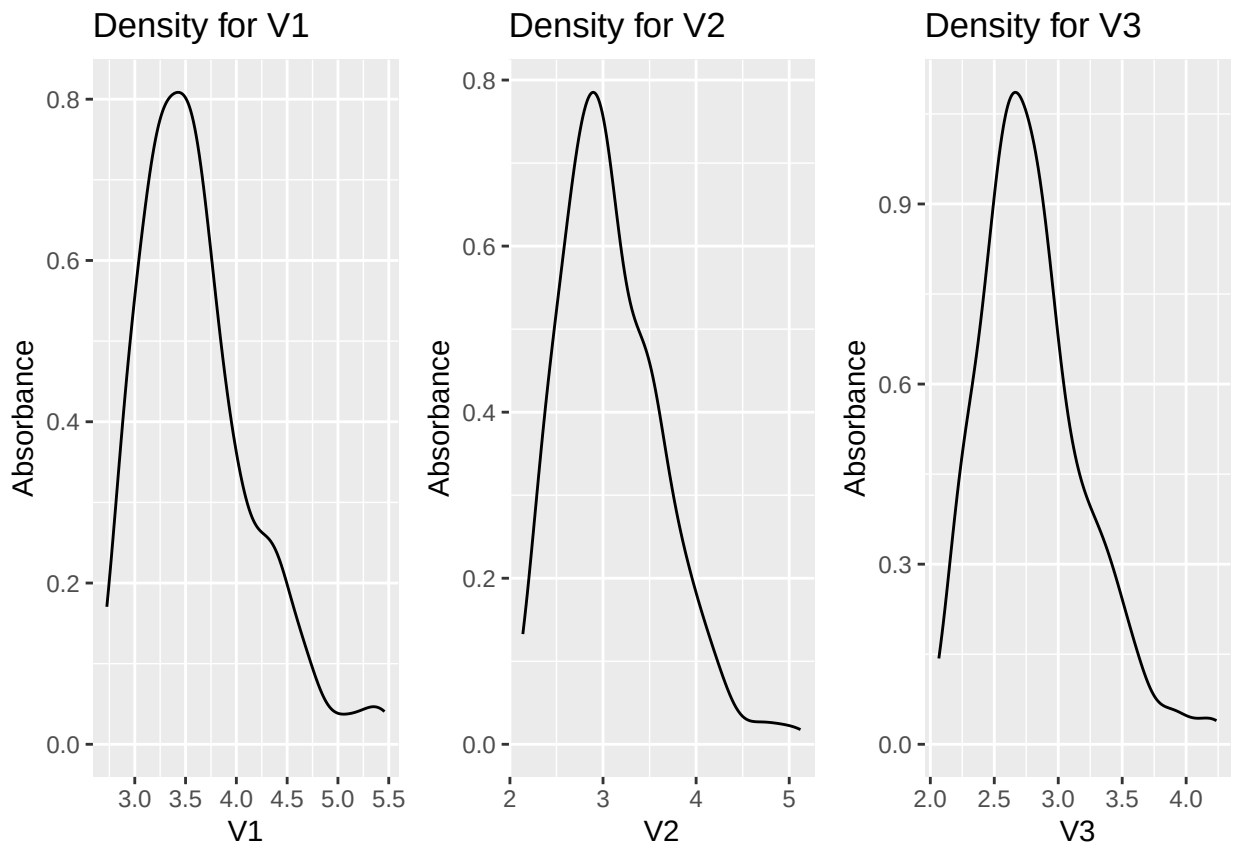
Based off of this random sample, we can anticipate strong correlation between most predictors, and hence we should apply PCA.

```
pcaObject <- prcomp(absorp, scale = TRUE)
varExplained <- pcaObject$sdev^2 / sum(pcaObject$sdev^2)
  qplot(x=c(1:10),y=varExplained[1:10])+
  geom_line() +
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 1)
```



Virtually all of the absorbance data can be explained by a single predictor. So let's now check some of the predictors' density graphs for pre-processing:

```
freqSample <- as.data.frame(absorp[,randP[1:3]])

v1 <- ggplot(freqSample, aes(x=V1)) + geom_density() + labs(
    title = "Density for V1",
    x = "V1", y = "Absorbance",
  )
v2 <- ggplot(freqSample, aes(x=V2)) + geom_density() + labs(
    title = "Density for V2",
    x = "V2", y = "Absorbance",
  )
v3 <- ggplot(freqSample, aes(x=V3)) + geom_density() + labs(
    title = "Density for V3",
    x = "V3", y = "Absorbance",
  )
ggarrange(v1, v2, v3, ncol = 3)
```



We see they're highly skewed, so we note for a Box-Cox transformation in the pre-processing.

Now for our modeling we have the following to-do list:

- Pre-processing predictors with a Box-Cox, scaling, and centering
- Split our data into train/test sets via 80:20
- Apply a 10-fold-CV for Ordinary LR, Partial Least Squares, and Ridge Regression

We also attempt the following settings individually for OLR:

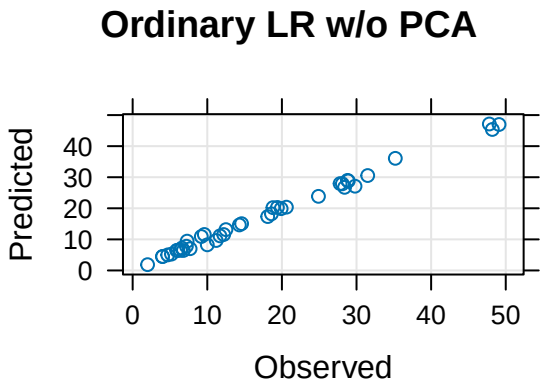
Feature extractions: PCA, and w/o PCA

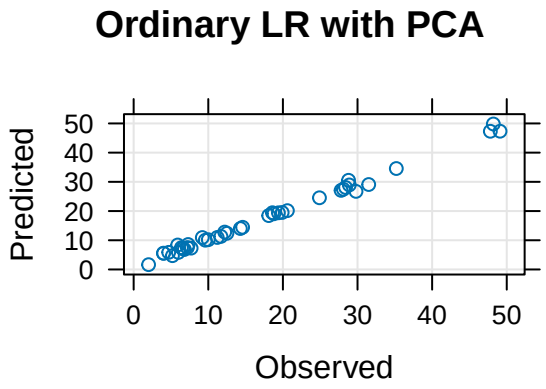
Ordinary Linear Regression

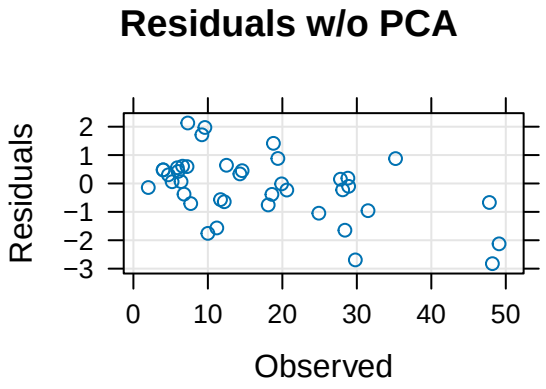
```
set.seed(1)
# Data splitting
trainingRows <- createDataPartition(endpoints[,1], p = 0.8, list = FALSE)

# Training set
trainAbs <- as.data.frame(absorp[trainingRows,])
trainFat <- as.data.frame(endpoints[trainingRows,2])
colnames(trainFat) <- "fatResponse"
combinedTrainingSet <- cbind(trainAbs, trainFat)

# Testing set
testAbs <- as.data.frame(absorp[-trainingRows,])
testFat <- as.data.frame(endpoints[-trainingRows,2])
```

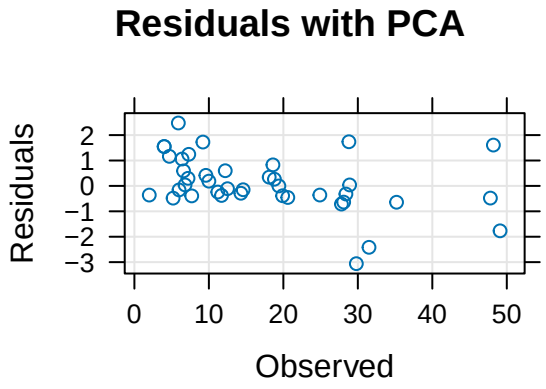
```r
colnames(testFat) <- "fatResponse"
combinedTestSet <- cbind(testAbs, testFat)

# 10-fold Cross-Validation parameter
ctrl <- trainControl(method = "cv", number = 10)

# OLR with PCA
OLRwPCA <- train(fatResponse ~ .,
                 data = combinedTrainingSet,
                 method = 'pcr',
                 preProcess = c("BoxCox", "center", "scale"),
                 tuneLength = 25,
                 trControl = ctrl
                 )
OLRwPCA_Results <-predict(OLRwPCA,combinedTestSet)

# OLR without PCA
OLRwoPCA <- train(fatResponse ~ .,
                 data = combinedTrainingSet,
                 method = 'lm',
                 preProcess = c("BoxCox", "center", "scale"),
                 trControl = ctrl
                 )
OLRwoPCA_Results <-predict(OLRwoPCA,combinedTestSet)
```

```r
# Setting up observations & residuals
observed <- combinedTestSet[,101]
residualswPCA <- OLRwPCA_Results-observed
residualswoPCA <- OLRwoPCA_Results-observed
# OLR & Observed vs Predicted Plots
plot1 <- xyplot(OLRwoPCA_Results ~ observed,
       type = c("p","g"),
       xlab = "Observed", ylab ="Predicted",
       main = "Ordinary LR w/o PCA"
       )

plot2 <- xyplot(OLRwPCA_Results ~ observed,
       type = c("p","g"),
       xlab = "Observed", ylab ="Predicted",
       main = "Ordinary LR with PCA"
       )
# Residual Plots
plot3 <- xyplot(residualswoPCA ~ observed,
       type = c("p","g"),
       xlab = "Observed", ylab ="Residuals",
       main = "Residuals w/o PCA"
       )
plot4 <- xyplot(residualswPCA ~ observed,
       type = c("p","g"),
       xlab = "Observed", ylab ="Residuals",
       main = "Residuals with PCA"
       )
grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)
```










```
# OLR with PCA Coefficient of Determination (R^2)
caret::R2(OLRwPCA_Results, observed)
```

```
## [1] 0.9935584
```

```
# OLR with PCA Root Mean Squared Error
caret::RMSE(OLRwPCA_Results, observed)
```

```
## [1] 1.081117
```

```
# OLR w/o PCA Coefficient of Determination (R^2)
caret::R2(OLRwoPCA_Results, observed)
```

```
## [1] 0.9937194
```

```
# OLR with PCA Root Mean Squared Error
caret::RMSE(OLRwoPCA_Results, observed)
```

```
## [1] 1.119854
```

We observe a much higher R^2, and $RMSE$ values in the OLR model w/o PCA than with PCA. Furthermore, the plots themselves are pretty telling of the predictive ability differences. Finally, the residuals are clearly non-normal in the PCA model so it violates LR assumptions. Hence, its fair to favor w/o PCA OLR over with PCA OLR.

However, we can't settle on w/o PCA OLR yet since we still have a ton of highly correlated predictors that we could still benefit from via a *supervised dimension-reduction* method: *PLS*.

Partial Least Squares

```
set.seed(1)
# PLS
PLS <-  train(fatResponse ~ .,
                  data = combinedTrainingSet,
                  method = "pls",
                  tuneLength = 20,
                  trControl = ctrl,
                  preProc = c("BoxCox", "center", "scale"))
PLS_Results <-predict(PLS,combinedTestSet)
# PLS Coefficient of Determination (R^2)
caret::R2(PLS_Results, observed)
```

```
## [1] 0.994109
```

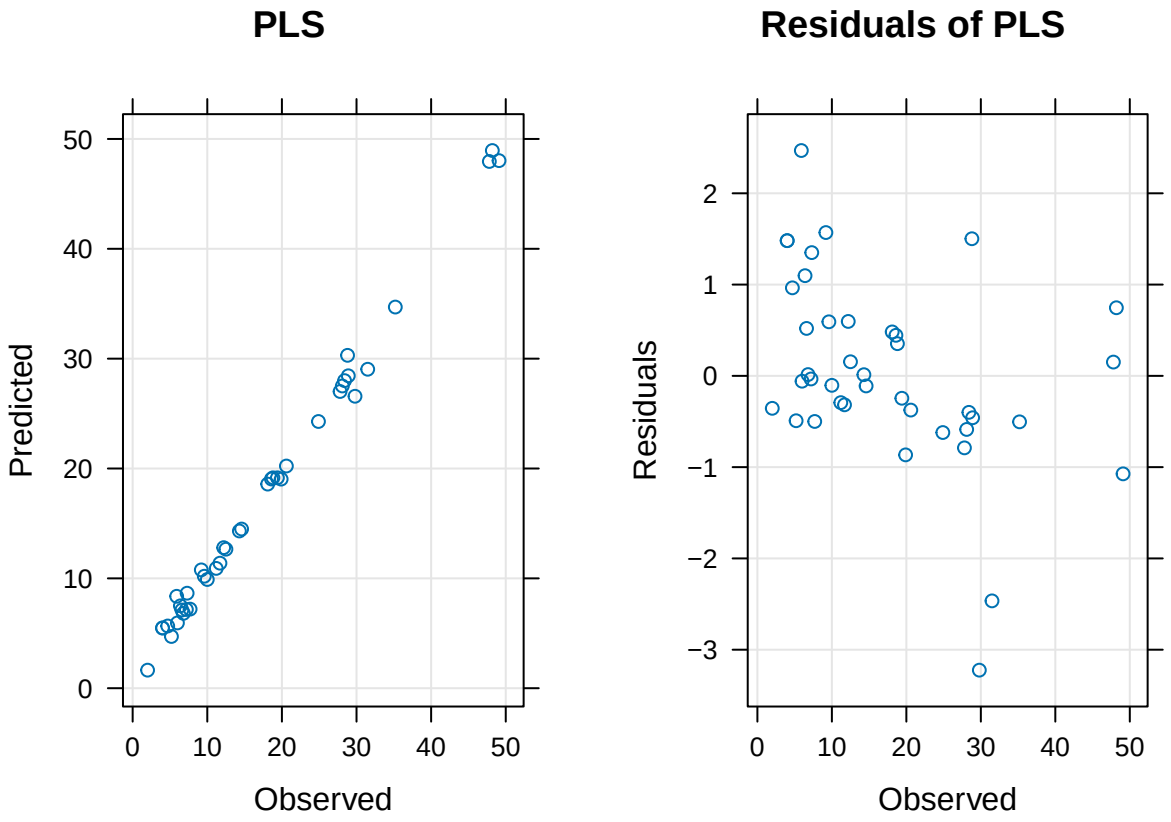
```
# PLS RMSE
caret::RMSE(PLS_Results, observed)
```

```
## [1] 1.032904
```

```
# PLS optimal number of components
PLS$bestTune
```

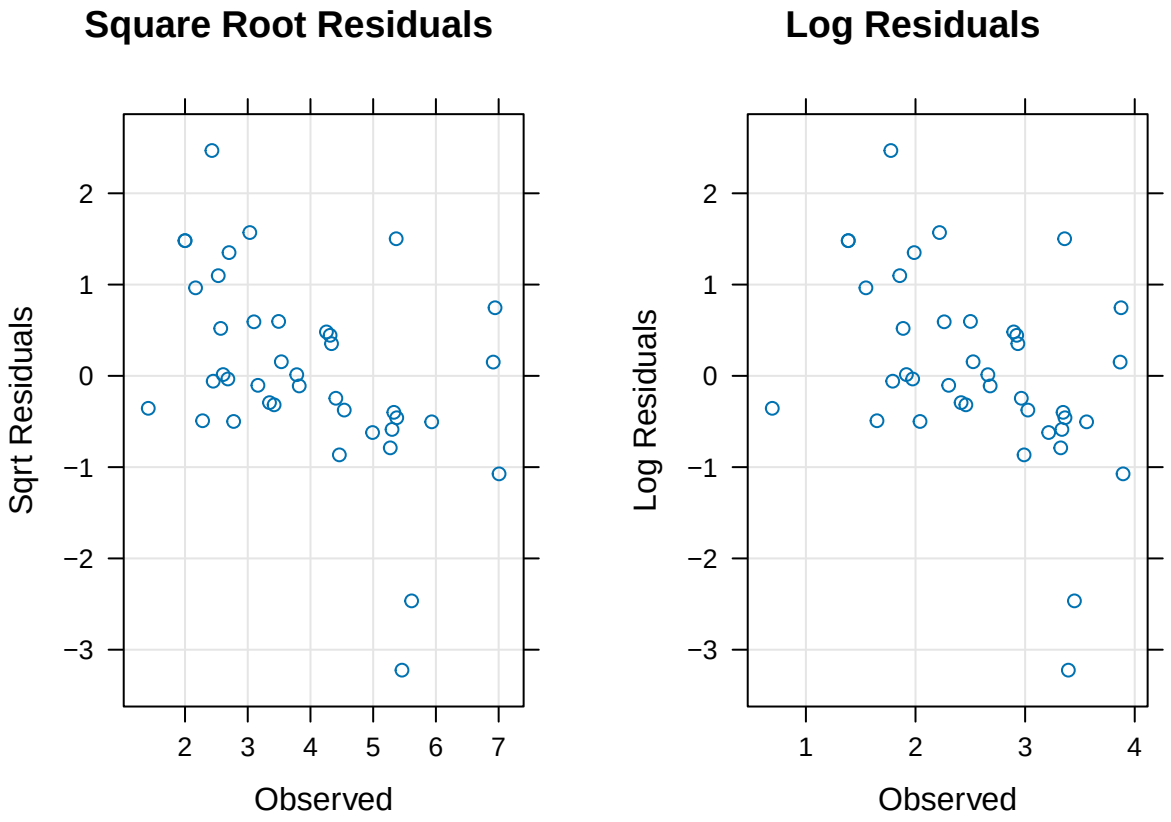
```
##    ncomp
## 19    19
```

```
# PLS Observed vs Predicted plot
residualsPLS <- PLS_Results - observed
plotPLS1 <- xyplot(PLS_Results ~ observed,
       type = c("p","g"),
       xlab = "Observed", ylab ="Predicted",
       main = "PLS"
       )
# PLS Residual plots
plotPLS2 <- xyplot(residualsPLS ~ observed,
       type = c("p","g"),
       xlab = "Observed", ylab ="Residuals",
       main = "Residuals of PLS"
       )
grid.arrange(plotPLS1, plotPLS2, ncol = 2)
```



PLS performs slightly better than OLR with larger R^2 and smaller $RMSE$. The predicted vs observed plot is strongly linear, but we do see some violation in the constant-variance assumption. We attempt some transformations of the response variable to save the assumption:

```
plotPLSTrans1 <- xyplot(residualsPLS ~ sqrt(observed),
       type = c("p","g"),
       xlab = "Observed", ylab ="Sqrt Residuals",
       main = "Square Root Residuals"
       )
plotPLSTrans2 <- xyplot(residualsPLS ~ log(observed),
       type = c("p","g"),
       xlab = "Observed", ylab ="Log Residuals",
       main = "Log Residuals"
       )
grid.arrange(plotPLSTrans1, plotPLSTrans2, ncol = 2)
```



Unfortunately the transformations still violate the constant-variance assumption. So we note this as a deficiency in the PLS model.

Ridge Regression

```
ridgeGrid <- data.frame(.lambda = seq(0, .1, length = 15))
set.seed(1)
ridgeReg <- train(fatResponse ~ .,
                  data = combinedTrainingSet,
                  method = "ridge",
                  tuneGrid = ridgeGrid,
                  trControl = ctrl,
                  preProc = c("BoxCox","center", "scale"))
ridgeRegResults <- predict(ridgeReg, combinedTestSet)
# Ridge  Coefficient of Determination (R^2)
caret::R2(ridgeRegResults, observed)
```

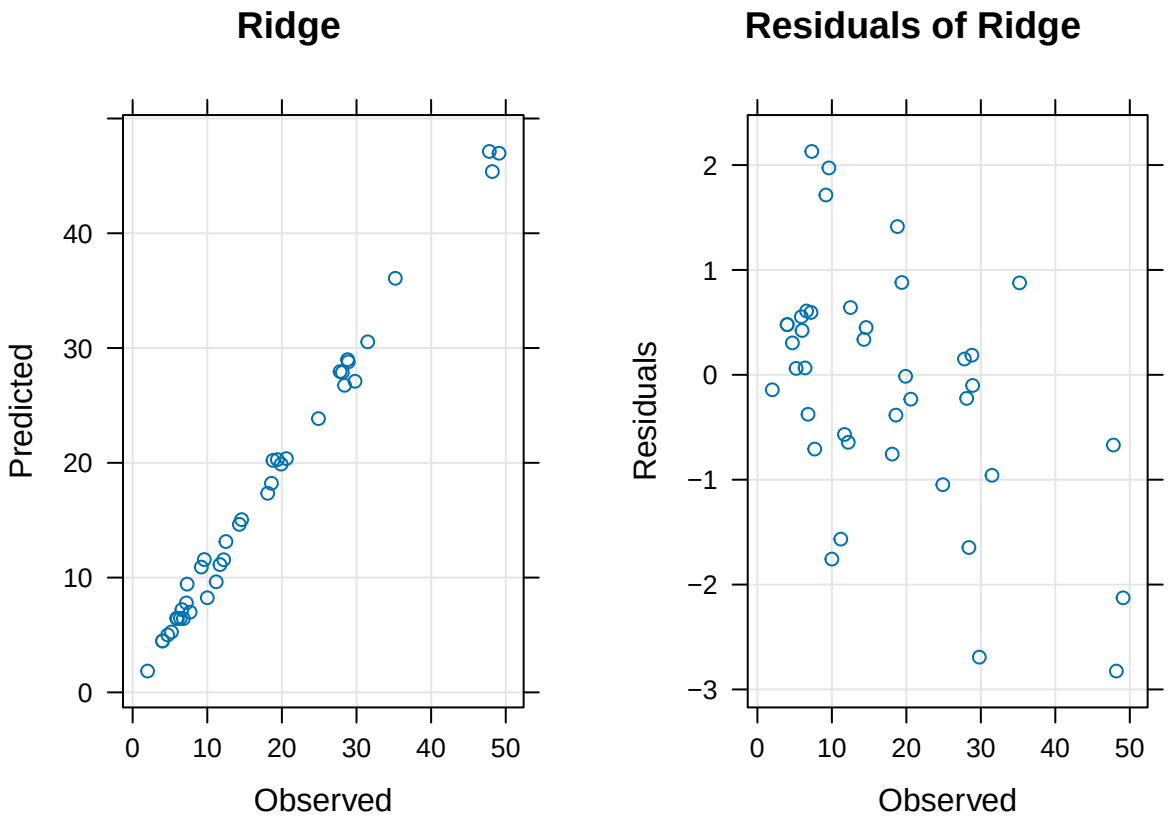
```
## [1] 0.9937194
```

```
# Ridge RMSE
caret::RMSE(ridgeRegResults, observed)
```

```
## [1] 1.119859
```

```
# Ridge Observed vs Predicted plot
residualsRidge <- ridgeRegResults - observed
plotRidge1 <- xyplot(ridgeRegResults ~ observed,
       type = c("p","g"),
```

```
       xlab = "Observed", ylab ="Predicted",
       main = "Ridge"
       )
# Ridge Residual plots
plotRidge2 <- xyplot(residualsRidge ~ observed,
       type = c("p","g"),
       xlab = "Observed", ylab ="Residuals",
       main = "Residuals of Ridge"
       )
grid.arrange(plotRidge1, plotRidge2, ncol = 2)
```



RR has similar R^2 and $RMSE$ values as the two preceding models; residuals are randomly distributed, and a strong linear relationship is evident in the predicted vs observed.

Since PLS has the largest R^2 value of 0.994, lowest $RMSE$ of 1.03, and uses only 19 independent predictors instead of the complete set of 100 absorbances, it's clearly the strongest model out of the three. However, it should be noted that other nonlinear models may perform better due to PLS's violation of the constant variance assumption.

Using Partial Least Squares, we can use 19 predictors to explain 99.4% of the variability in fat content of meat, with an RMSE of 1.03.