# ShovanBiswas-DATA24_Homework_07

Shovan Biswas

10/30/2020

## Libraries

```
library(tidyverse)
library(kableExtra)
library(corrplot)
library(reshape2)
library(caret)
library(Amelia)
library(dlookr)
library(fpp2)
library(plotly)
library(gridExtra)
library(readxl)
library(ggplot2)
library(urca)
library(tseries)
library(AppliedPredictiveModeling)
library(RANN)
library(psych)
library(e1071)
library(corrplot)
library(glmnet)
```

## Exercise 6.2

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 1,107 binary molecular predictors for the 165 compounds, while permeability contains permeability response.

```
data(permeability)
```

(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?

```
dim(fingerprints)
```

```
## [1]  165 1107
```

There are 1107 predictors and 165 observations.

We'll filter out the predictors having low frequenies, with nearzero function.

```
fingerprints_filtrate <- fingerprints[, -nearZeroVar(fingerprints)]
dim(fingerprints_filtrate)
```

```
## [1] 165 388
```

There are 388 predictors left for modeling. 719 columns were removed.

At this point, I'll make a quick check on the data, whether it contains NA data.

```
any(is.na(fingerprints_filtrate))
```

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

The filtrate data looks good.

(c) Split the data into a training and a test set, pre-process the data, and tune a PLS model. How many latent variables are optimal and what is the corresponding resampled estimate of R2?

First we'll pre-process the data. In order to pre-process the data, we'll follow the guidlines in the text book on page 105, 2nd para. So, we'll try to drive out highly correlated columns. So, we'll do a pai-wise analysis. We'll use the correlation value of 0.9, used in the book.

```
highCorr <- findCorrelation(cor(fingerprints_filtrate), 0.90)

fingerprints_filtrate_2 <- fingerprints_filtrate[, -highCorr]

dim(fingerprints_filtrate_2)
```

```
## [1] 165 110
```

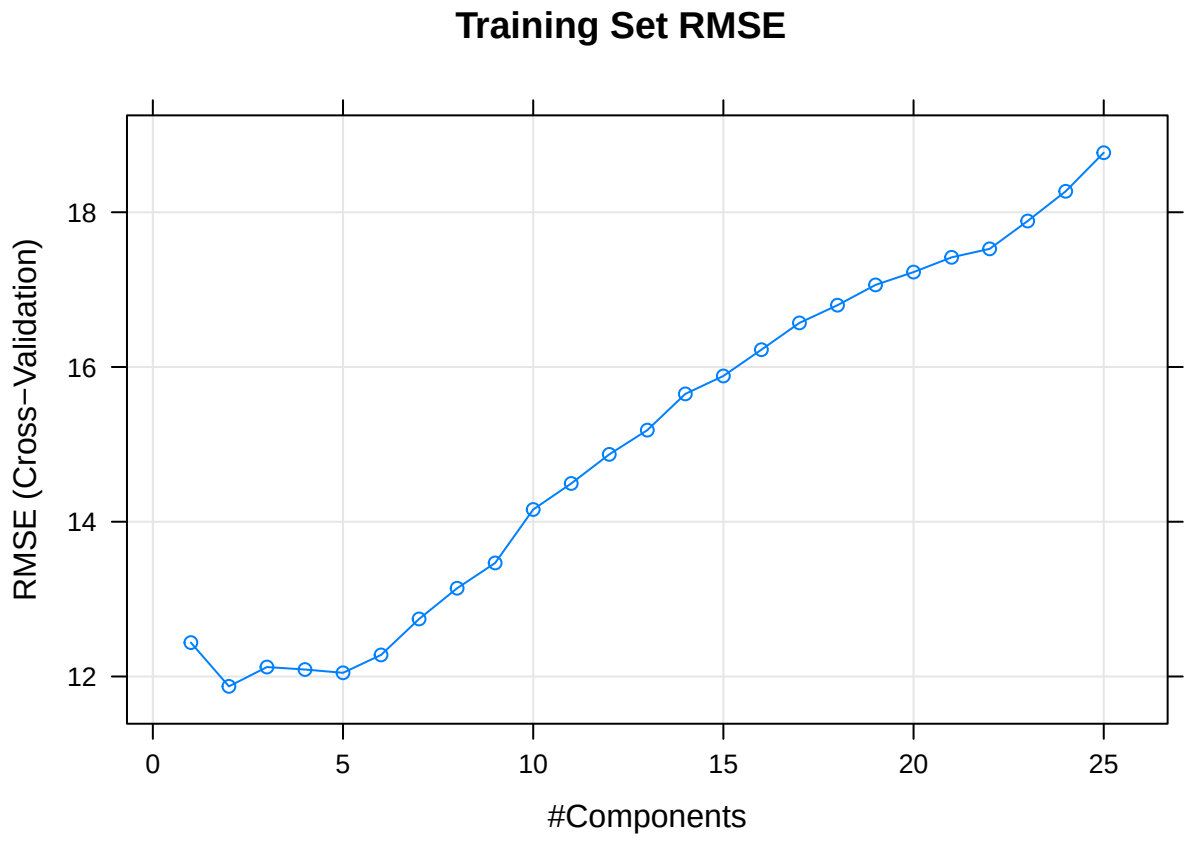
In the pair-wise analysis, eventually 278 highly correlated predictors were removed.

Now, we're in a position to split the data in to train (0.7) and test (0.3). After splitting, I'll run the PLS model.

```r
set.seed(123)

split_index <- createDataPartition(permeability, p = 0.7, list = FALSE)
X_train <- fingerprints_filtrate_2[split_index, ]
y_train <- permeability[split_index, ]
X_test <- fingerprints_filtrate_2[-split_index, ]
y_test <- permeability[-split_index, ]
```

I'll do PLS model.

```r
pls_model <- train(x = X_train, y = y_train, method = "pls", preProc = c("center", "scale"), trControl =

plot(pls_model, main = "Training Set RMSE")
```




```r
pls_model$results %>% filter(ncomp == pls_model$bestTune$ncomp) %>% select(ncomp, RMSE, Rsquared)
```

```
##   ncomp     RMSE  Rsquared
## 1     2 11.87233 0.4552083
```

2 latent variables are optimal and this catches 45% of the permeability.

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

```
pls_prediction <- predict(pls_model, newdata = X_test)

results <- data.frame(Model = "PLS", RMSE = caret::RMSE(pls_prediction, y_test), Rsquared = caret::R2(pl
results
```

```
##   Model     RMSE  Rsquared
## 1   PLS 10.71883 0.4475969
```

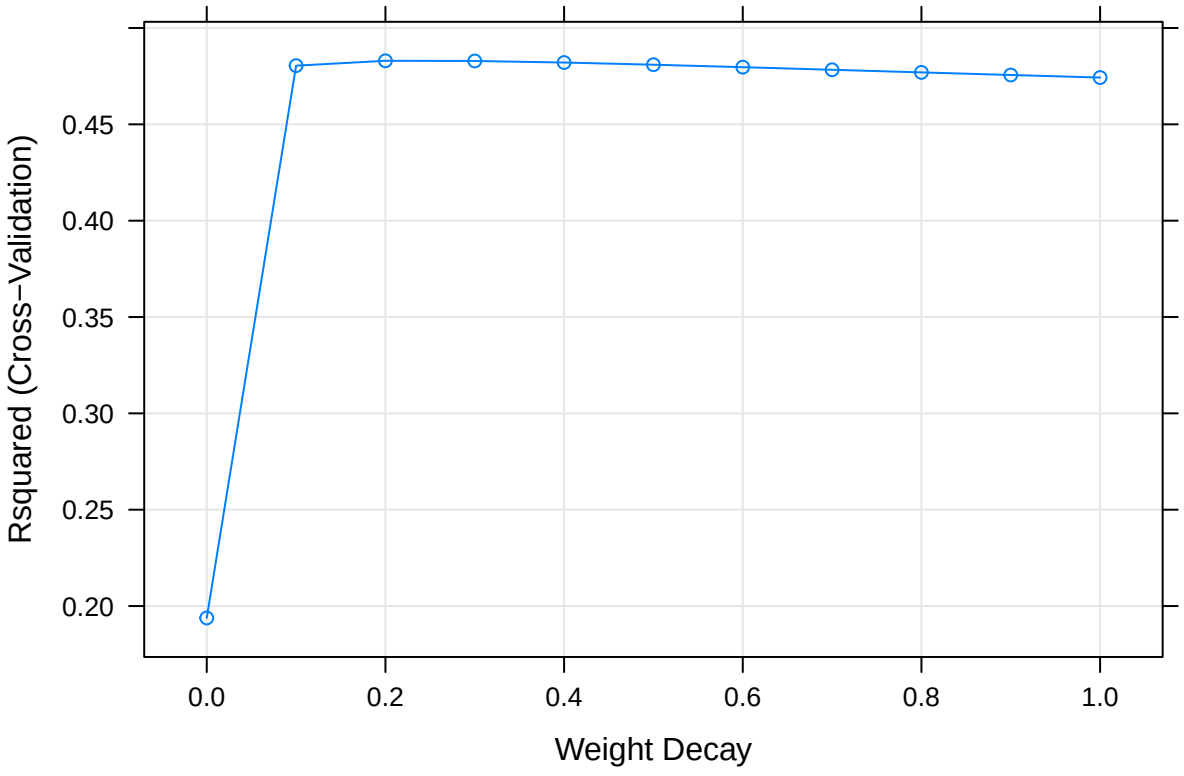
(e) Try building other models discussed in this chapter. Do any have better predictive performance?

We'll do Ridge, Lasso and Elastic net regressions.

## Ridge Regression

```
ridge_fitting <- train(x = X_train, y = y_train, method = 'ridge', metric = 'Rsquared', tuneGrid = data
                       preProcess = c('center', 'scale'))
```

```
plot(ridge_fitting)
```



```
ridge_predictions <- predict(ridge_fitting, newdata = X_test)

ridge_results <- data.frame(Model = "Ridge Regression", RMSE = caret::RMSE(ridge_predictions, y_test), R

ridge_results
```

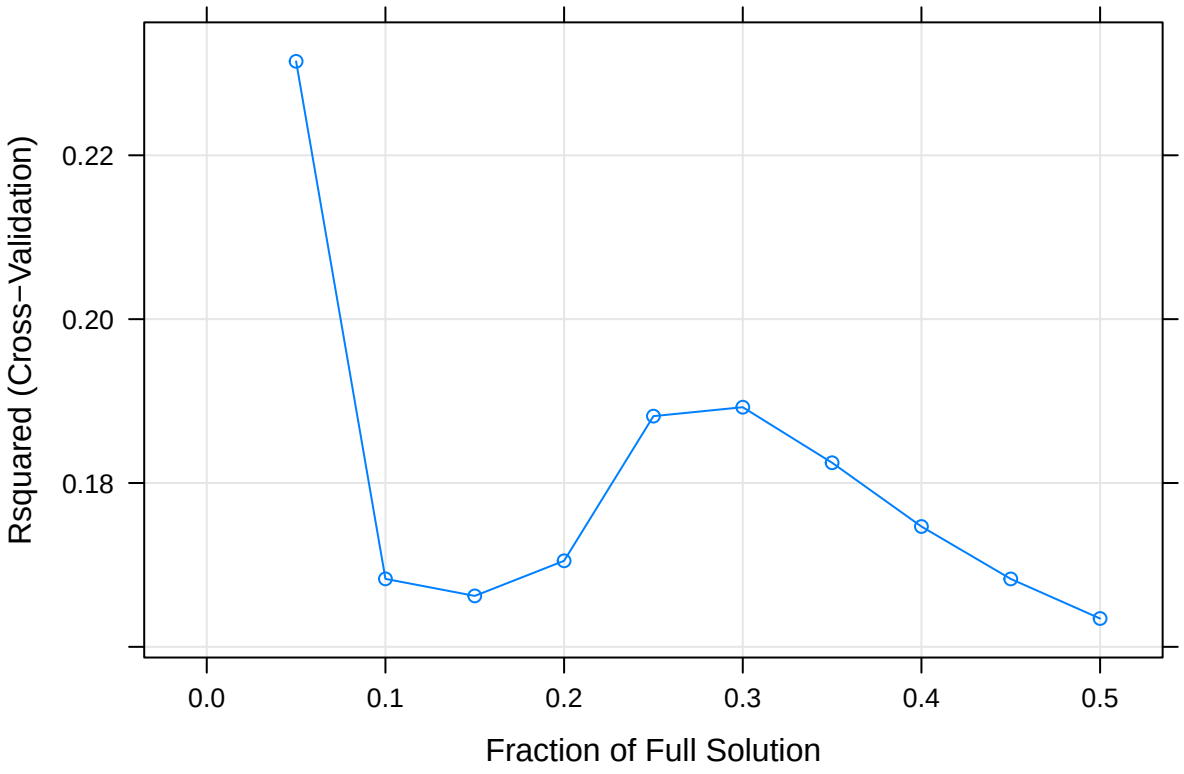
```
##              Model     RMSE  Rsquared
## 1 Ridge Regression 10.64581 0.5395415
```

## Lasso Regression

```
lasso_fitting <- train(x = X_train, y = y_train, method ='lasso', metric = 'Rsquared', tuneGrid = data.f
                  preProcess = c('center', 'scale'))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.

## Warning in train.default(x = X_train, y = y_train, method = "lasso", metric =
## "Rsquared", : missing values found in aggregated results
```

```
plot(lasso_fitting)
```



```
lasso_predictions <- predict(lasso_fitting, newdata = X_test)

lasso_results <- data.frame(Model = "Lasso Regression", RMSE = caret::RMSE(lasso_predictions, y_test), R
lasso_results
```

```
##              Model     RMSE  Rsquared
## 1 Lasso Regression 12.98478 0.4318239
```

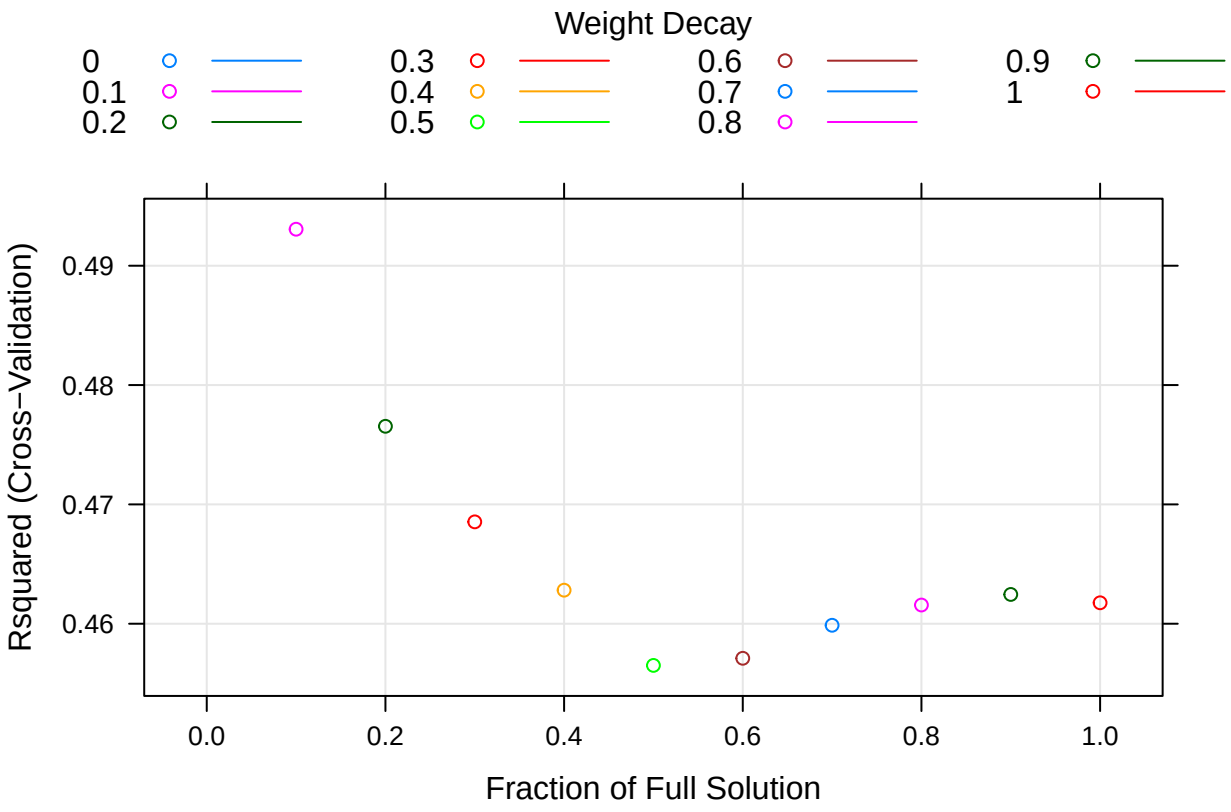
## Elastic Net Regression

```
elastic_fitting <- train(x = X_train, y = y_train, method = 'enet', metric = 'Rsquared', tuneGrid = data
                 trControl = trainControl(method = 'cv'), preProcess = c('center', 'scale'))
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
## Warning in train.default(x = X_train, y = y_train, method = "enet", metric =
## "Rsquared", : missing values found in aggregated results
```

```
plot(elastic_fitting)
```



```
elastic_predictions <- predict(elastic_fitting, newdata = X_test)

elastic_results <- data.frame(Model = "Elastic Net Regression", RMSE = caret::RMSE(elastic_predictions,

elastic_results
```

```
##                    Model     RMSE  Rsquared
## 1 Elastic Net Regression 11.43583 0.3436236
```

There is no improvement compared to PLS.

**Summary**

```
pls_model$results %>%
  filter(ncomp == pls_model$bestTune$ncomp) %>%
  mutate("Model" = "PLS") %>%
  select(Model, RMSE, Rsquared) %>%
  as.data.frame() %>%
  bind_rows(ridge_results) %>%
  bind_rows(lasso_results) %>%
  bind_rows(elastic_results) %>%
  arrange(desc(Rsquared))
```

```
##                    Model     RMSE  Rsquared
## 1       Ridge Regression 10.64581 0.5395415
## 2                    PLS 11.87233 0.4552083
## 3       Lasso Regression 12.98478 0.4318239
## 4 Elastic Net Regression 11.43583 0.3436236
```

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

The two close candidates for consideration are PLS and Elastic Regression. But, in Ealstic, the although RMSE is lower, the R-squared is further from zero. So, considering RMSE and R-squared, PLS seems to be the better candidate.

## Exercise 6.3

6.3. 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(chemicalManufacturing)
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.

Note: The PDF version of the book has chemicalManufacturing, which didn't work. The paper copy has chemicalManufacturingProcess, which also didn't work. Tried ChemicalManufacturingProcess, which worked.

```
data(ChemicalManufacturingProcess)
dim(ChemicalManufacturingProcess)
```

```
## [1] 176  58
```

There are 176 observations and 57 predictors, out of which 12 are biological materials and 45 are manufacturing process. Additionally, there is a response variable Yield.

(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).

```
table(is.na(ChemicalManufacturingProcess))
```

```
##
## FALSE  TRUE
## 10102   106
```

We observed in (a), above, that there are data elements (i.e. $row \cap columns$). Out of these, NA data elements have NA. Now, I'll impute the data.

```
# data_imputed <- preProcess(ChemicalManufacturingProcess, 'knnImpute')
# data_imputed <- predict(data_imputed, ChemicalManufacturingProcess)
#
# data_imputed <- preProcess(ChemicalManufacturingProcess, method = c('bagImpute'))
# data_imputed <- predict(data_imputed, ChemicalManufacturingProcess)
#
data_imputed <- mice(ChemicalManufacturingProcess, m = 1, method = "pmm", print = F) %>% complete()
```

```
## Warning: Number of logged events: 135
```

I checked the results of impute with couple of other methods, namely *knnImpute* and *bagImpute*, but observed that while *knnImpute* changed proper data to something else, *bagImpute* only imputed NA values, which is what I wanted. However, *bagImpute* took quite long time to process, so I opted for *mice()* to impute the NA data.

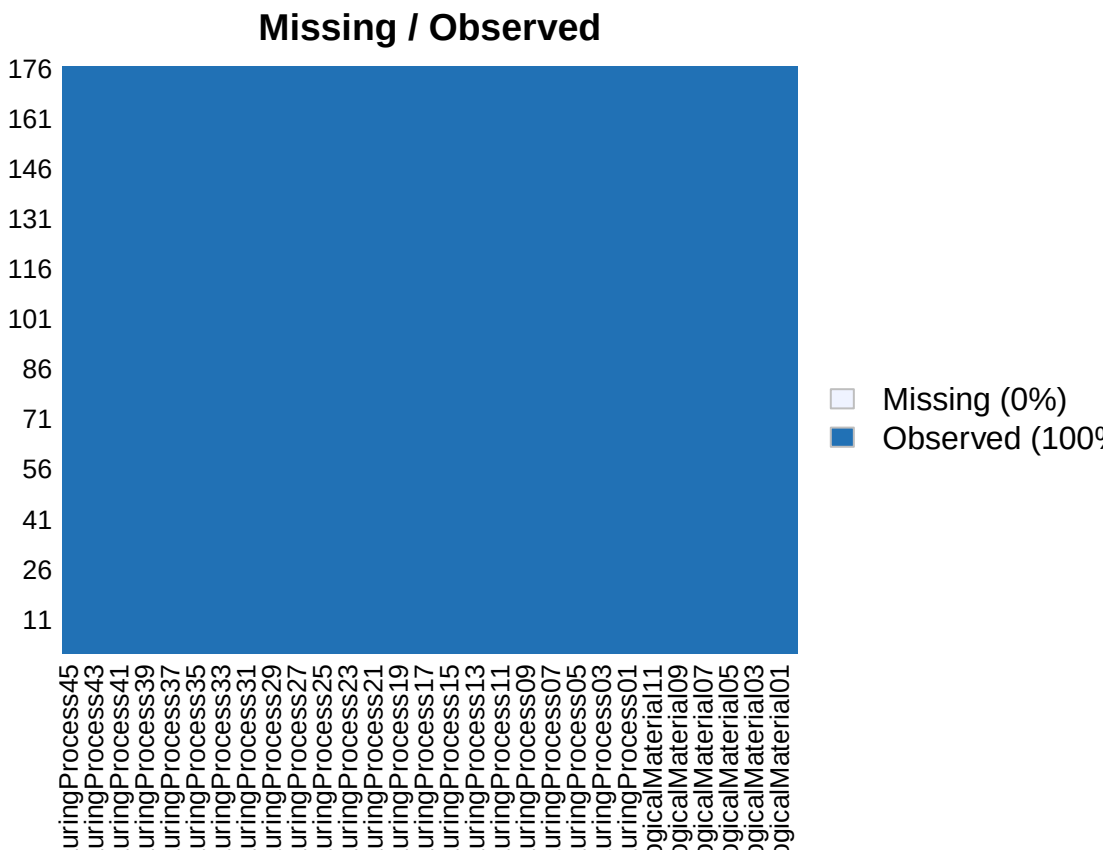
Now, on checking for NA again, it looks clean.

```
any(is.na(data_imputed))
```

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

A visulization of the clean data.

```
data_imputed %>% missmap(main = "Missing / Observed")
```

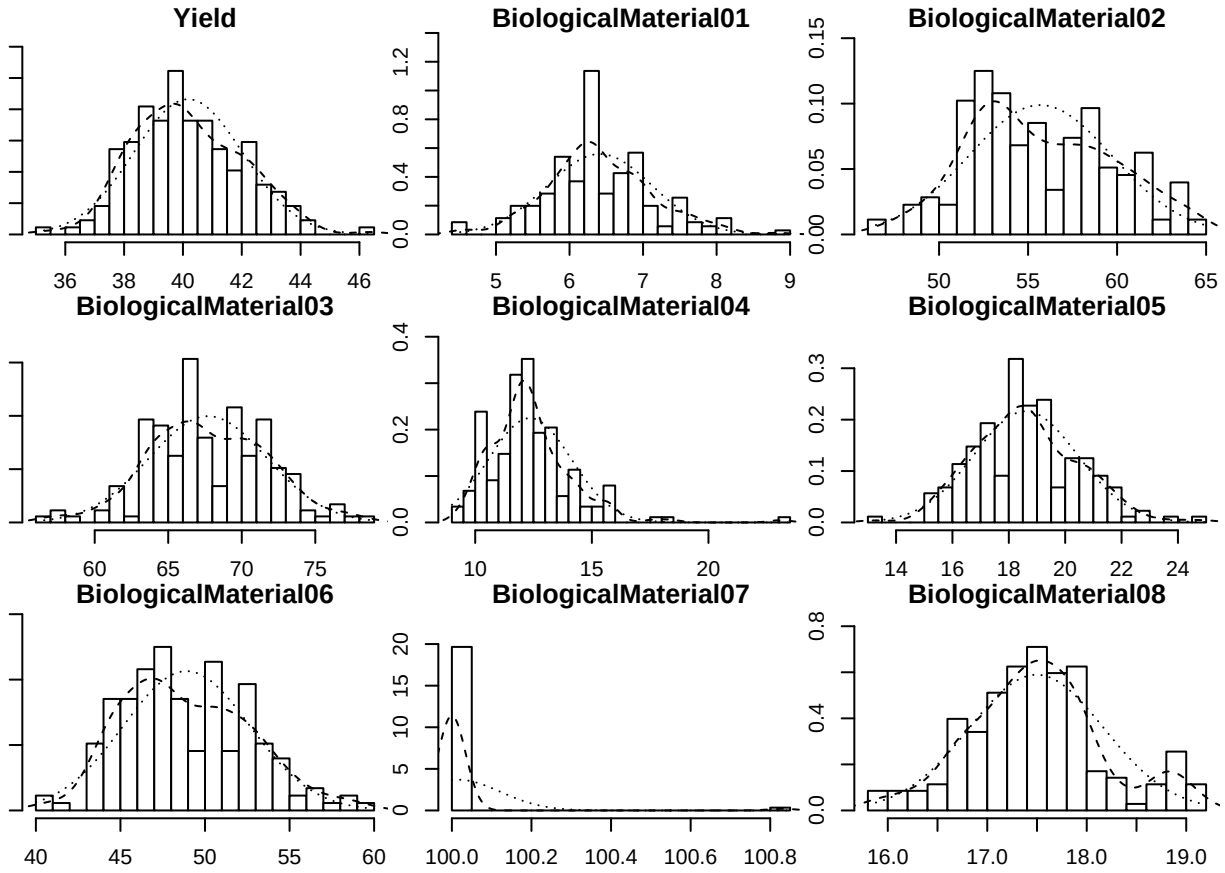


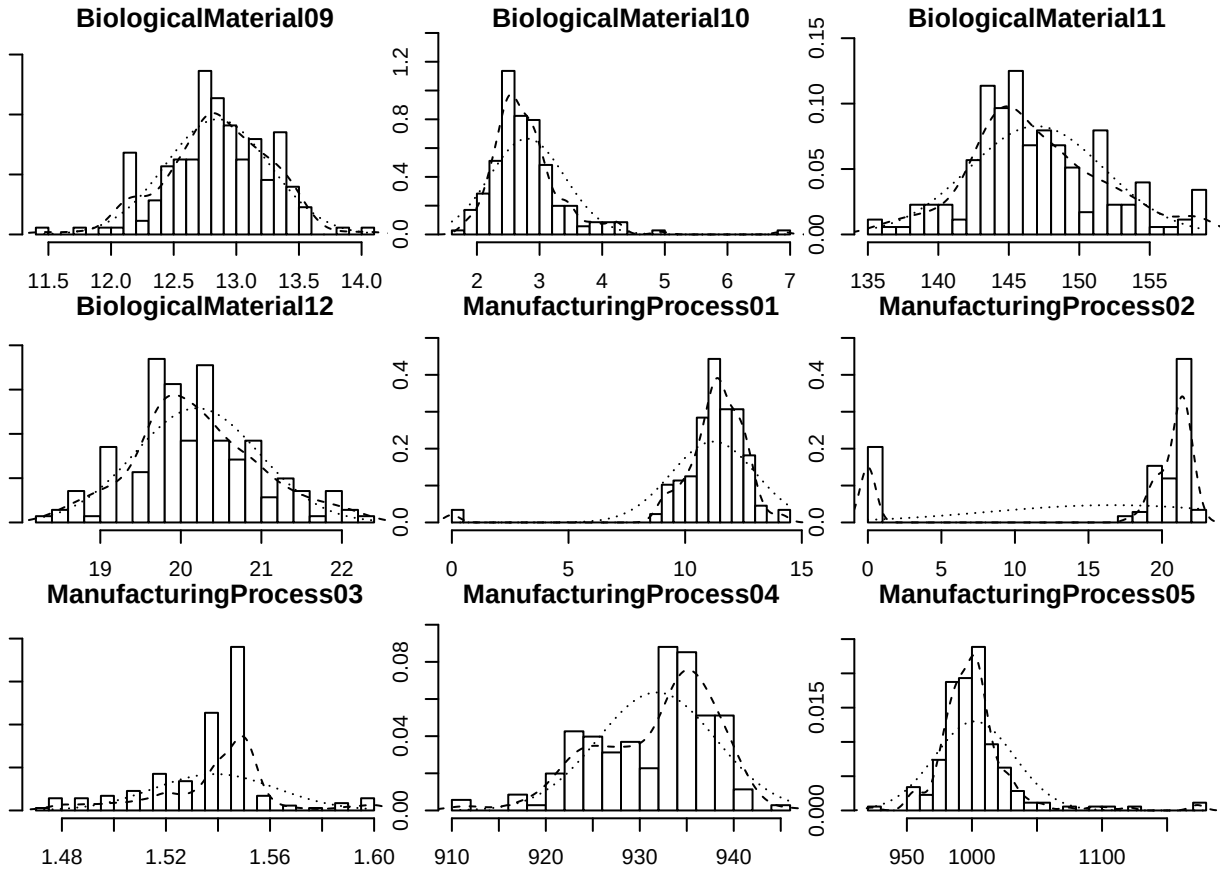

(c) Split the data into a training and a test set, pre-process the data, and tune a model of your choice from this chapter. What is the optimal value of the performance metric?

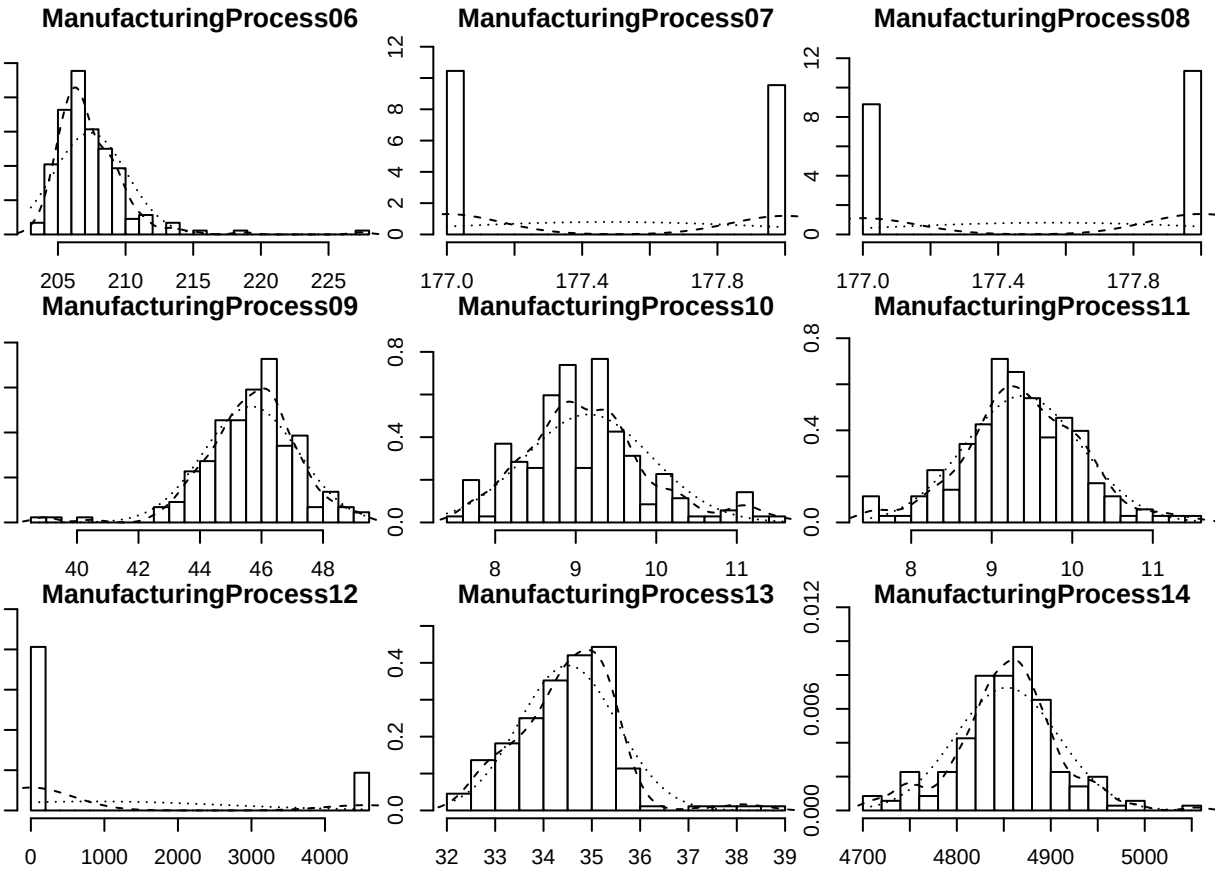
First we'll pre-process the data. In order to pre-process the data, we'll follow the guidlines in the text book on page 105, 2nd para. So, first we'll try to drive out the skeness in the data, by BoxCox transformation.

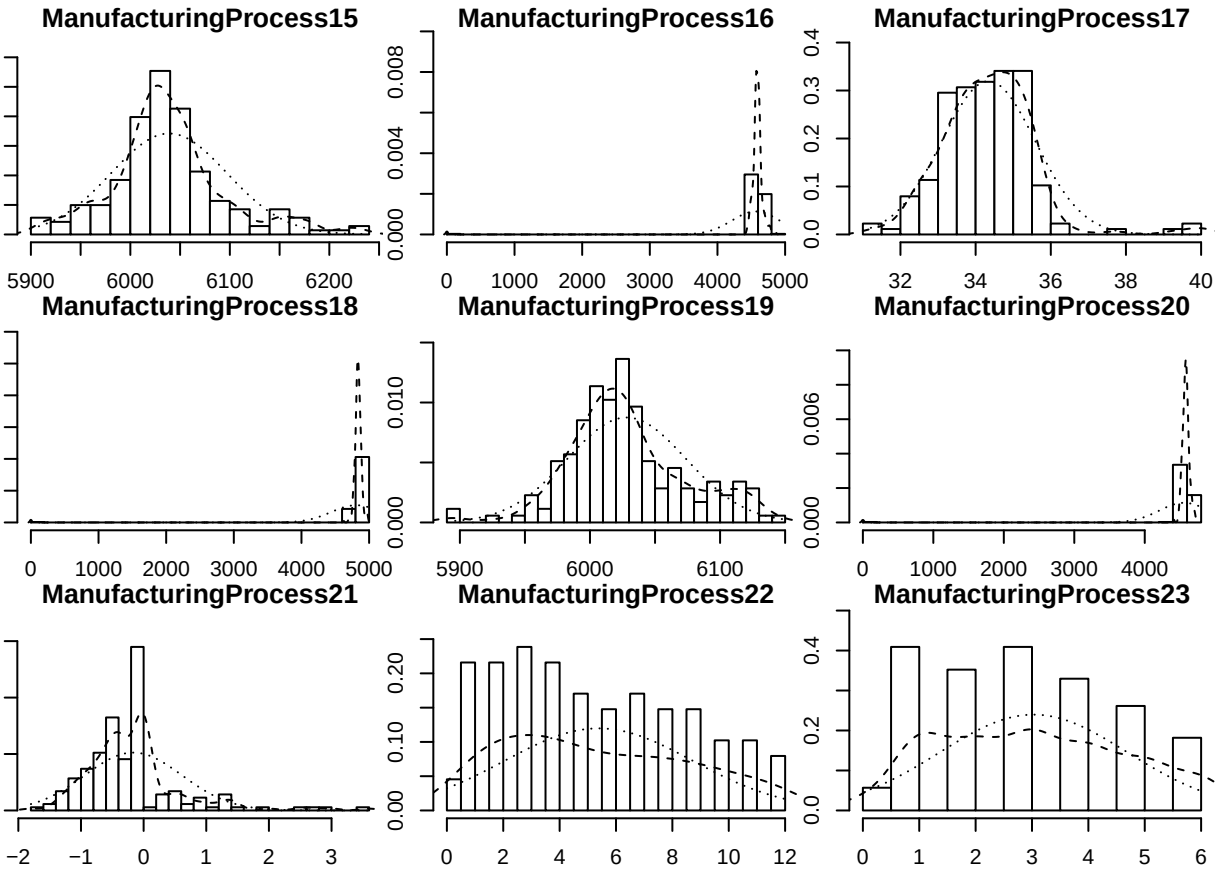
In order to determine the skewed columns, let's first get the histograms of the columns.

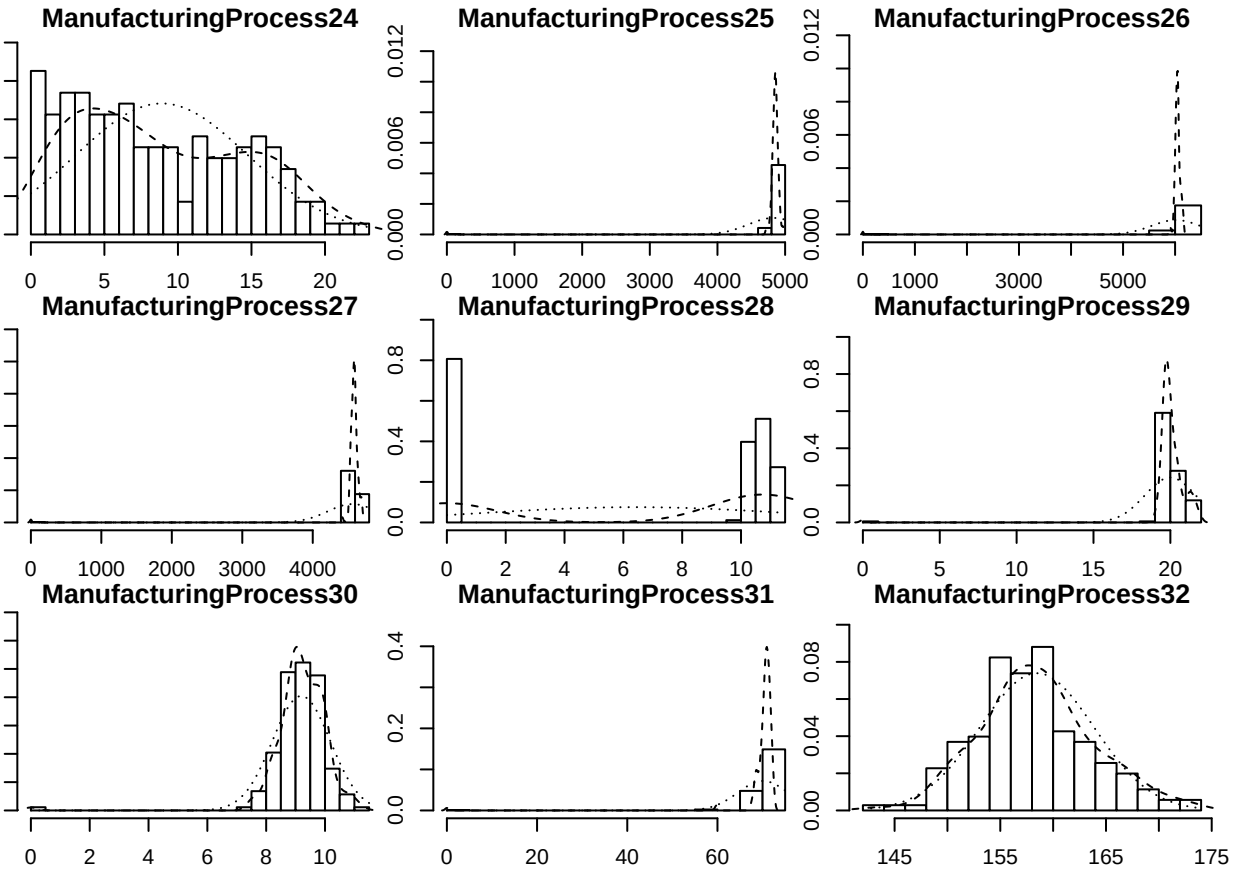
```
for(i in seq(from = 1, to = length(data_imputed), by = 9)) {
  if(i <= 54) {
    multi.hist(data_imputed[i:(i + 8)])
  } else {
    multi.hist(data_imputed[i:(i + 3)])
  }
}
```

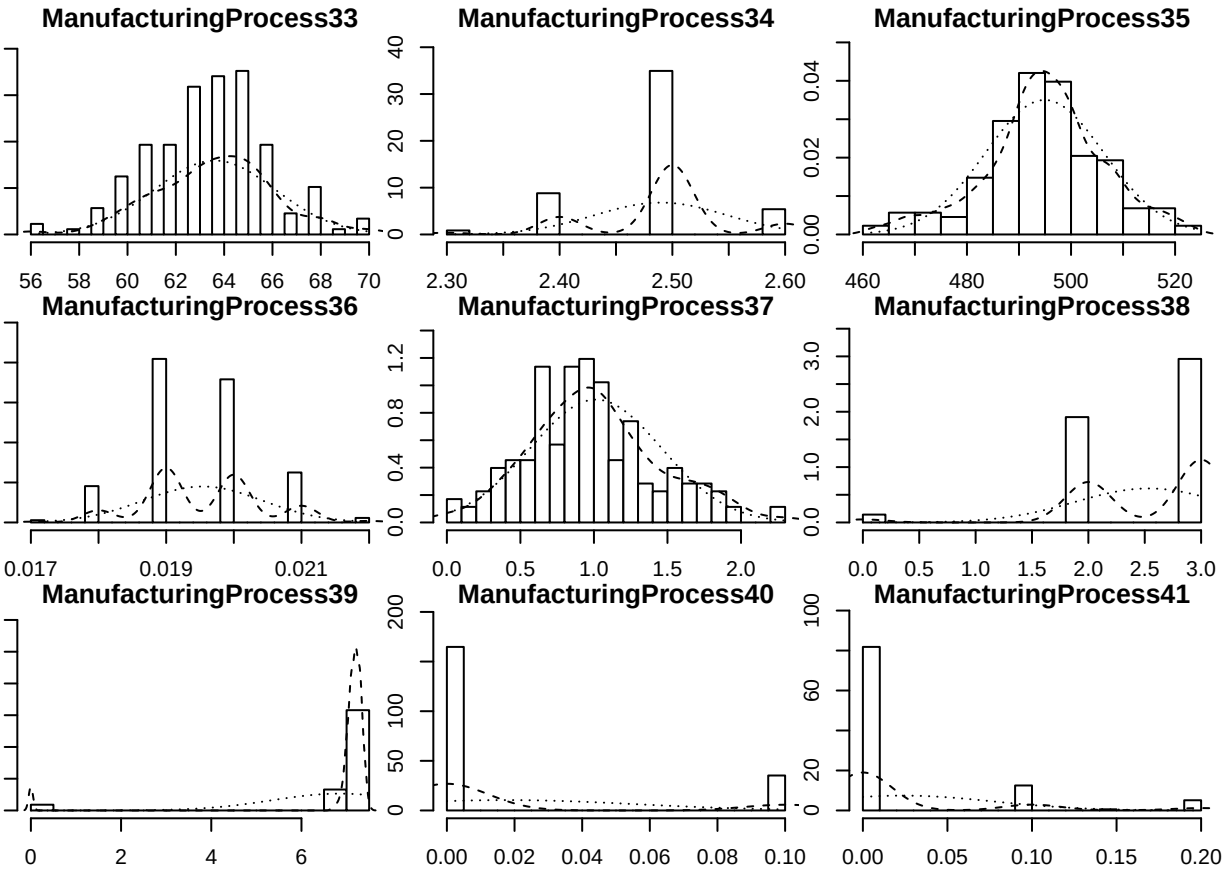

Yield
BiologicalMaterial01
BiologicalMaterial02
BiologicalMaterial03
BiologicalMaterial04
BiologicalMaterial05
BiologicalMaterial06
BiologicalMaterial07
BiologicalMaterial08

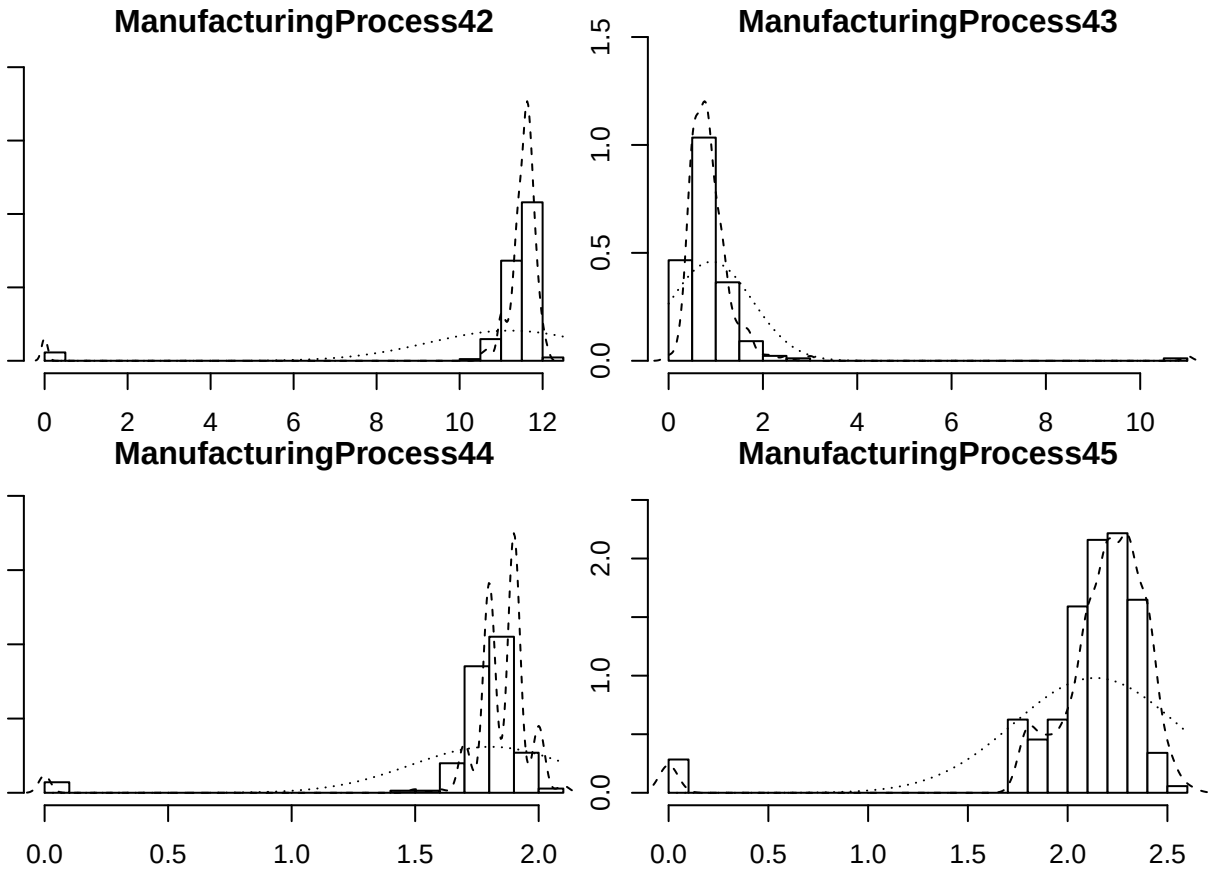

BiologicalMaterial09
11.5 12.0 12.5 13.0 13.5 14.0
BiologicalMaterial10
0.0 0.4 0.8 1.2
2 3 4 5 6 7
BiologicalMaterial11
0.00 0.05 0.10 0.15
135 140 145 150 155
BiologicalMaterial12
0.0 0.2 0.4
19 20 21 22
ManufacturingProcess01
0.0 0.2 0.4
0 5 10 15
ManufacturingProcess02
0.0 0.2 0.4
0 5 10 15 20
ManufacturingProcess03
0.00 0.04 0.08
1.48 1.52 1.56 1.60
ManufacturingProcess04
910 920 930 940
ManufacturingProcess05
0.000 0.015
950 1000 1100


ManufacturingProcess06
ManufacturingProcess07
ManufacturingProcess08
ManufacturingProcess09
ManufacturingProcess10
ManufacturingProcess11
ManufacturingProcess12
ManufacturingProcess13
ManufacturingProcess14


ManufacturingProcess15
ManufacturingProcess16
ManufacturingProcess17
ManufacturingProcess18
ManufacturingProcess19
ManufacturingProcess20
ManufacturingProcess21
ManufacturingProcess22
ManufacturingProcess23


ManufacturingProcess24
ManufacturingProcess25
ManufacturingProcess26
ManufacturingProcess27
ManufacturingProcess28
ManufacturingProcess29
ManufacturingProcess30
ManufacturingProcess31
ManufacturingProcess32


ManufacturingProcess33
ManufacturingProcess34
ManufacturingProcess35
ManufacturingProcess36
ManufacturingProcess37
ManufacturingProcess38
ManufacturingProcess39
ManufacturingProcess40
ManufacturingProcess41




Observing the above histograms, I decided the critical skewness, needing BoxCox transformation, to be 1 or higher. Based on this critical value, I am creating a vector transform_cols, which'll contain the column names of skewed columns.

The columns, whose skewness exceeds the critical value of 1, are printed below.

```
transform_cols <- c()

for(i in seq(from = 1, to = length(data_imputed), by = 1)) {
  if(abs(skewness(data_imputed[, i])) >= 1) {
    transform_cols <- append(transform_cols, names(data_imputed[i]))
    print(paste0(names(data_imputed[i]), ": ", skewness(data_imputed[, i])))
  }
}
```

```
## [1] "BiologicalMaterial04: 1.73231529612716"
## [1] "BiologicalMaterial07: 7.39866415111903"
## [1] "BiologicalMaterial10: 2.40237832446776"
## [1] "ManufacturingProcess01: -3.91982976238802"
## [1] "ManufacturingProcess02: -1.41478809390956"
## [1] "ManufacturingProcess05: 2.53033494973367"
## [1] "ManufacturingProcess06: 3.05894667993104"
## [1] "ManufacturingProcess12: 1.58765365596097"
## [1] "ManufacturingProcess16: -12.4202247618104"
## [1] "ManufacturingProcess17: 1.16297153272138"
## [1] "ManufacturingProcess18: -12.7361377794098"
```

```
## [1] "ManufacturingProcess20: -12.6383267799381"
## [1] "ManufacturingProcess21: 1.72911402860041"
## [1] "ManufacturingProcess25: -12.7793154414265"
## [1] "ManufacturingProcess26: -12.8290108756301"
## [1] "ManufacturingProcess27: -12.6990133622734"
## [1] "ManufacturingProcess29: -10.1244010540032"
## [1] "ManufacturingProcess30: -4.52141327621931"
## [1] "ManufacturingProcess31: -11.5730865773941"
## [1] "ManufacturingProcess38: -1.68180523930997"
## [1] "ManufacturingProcess39: -4.26912142852766"
## [1] "ManufacturingProcess40: 1.68588549698972"
## [1] "ManufacturingProcess41: 2.1783650588771"
## [1] "ManufacturingProcess42: -5.45000823228715"
## [1] "ManufacturingProcess43: 9.05487467577087"
## [1] "ManufacturingProcess44: -4.9703551700542"
## [1] "ManufacturingProcess45: -4.07794105369906"
```

Many of these histograms are skewed. So, following the recommendations of the text book (page 105, 2nd para), I'll apply Box-Cox transformation to remove the skewness.

```
lambda <- NULL
data_imputed_2 <- data_imputed

for (i in 1:length(transform_cols)) {
  lambda[transform_cols[i]] <- BoxCox.lambda(abs(data_imputed[, transform_cols[i]]))

  data_imputed_2[c(transform_cols[i])] <- BoxCox(data_imputed[transform_cols[i]], lambda[transform_cols
}
```

Now, we don't need to observe the histograms all over again. It will suffice to see the skewness.

We observe that skewness of most or all of the columns reduced and some even reduced to less than 1.

```
for(i in seq(from = 1, to = length(data_imputed_2), by = 1)) {
  if(abs(skewness(data_imputed_2[, i])) >= 1) {
    print(paste0(names(data_imputed_2[i]), ": ", skewness(data_imputed_2[, i])))
  }
}
```

```
## [1] "BiologicalMaterial07: 7.39866415111915"
## [1] "ManufacturingProcess01: -1.26052235841779"
## [1] "ManufacturingProcess02: -1.41478810473402"
## [1] "ManufacturingProcess05: 1.94224024604565"
## [1] "ManufacturingProcess06: 2.69669944331699"
## [1] "ManufacturingProcess12: 1.58765365596097"
## [1] "ManufacturingProcess16: -10.7847613325053"
## [1] "ManufacturingProcess18: -11.8943041042034"
## [1] "ManufacturingProcess20: -11.543922540353"
## [1] "ManufacturingProcess25: -12.0446761185964"
## [1] "ManufacturingProcess26: -12.2252236333278"
## [1] "ManufacturingProcess27: -11.756966256326"
## [1] "ManufacturingProcess29: -5.36166249032794"
## [1] "ManufacturingProcess31: -8.95417981700199"
## [1] "ManufacturingProcess39: -4.26912144579668"
```

```
## [1] "ManufacturingProcess40: 1.68588549698972"
## [1] "ManufacturingProcess41: 1.63587403403708"
## [1] "ManufacturingProcess42: -4.98855795039927"
## [1] "ManufacturingProcess43: -13.0412208841127"
## [1] "ManufacturingProcess44: -3.60942641714785"
## [1] "ManufacturingProcess45: -2.13833528385902"
```

Now, I'll split data into train and test.

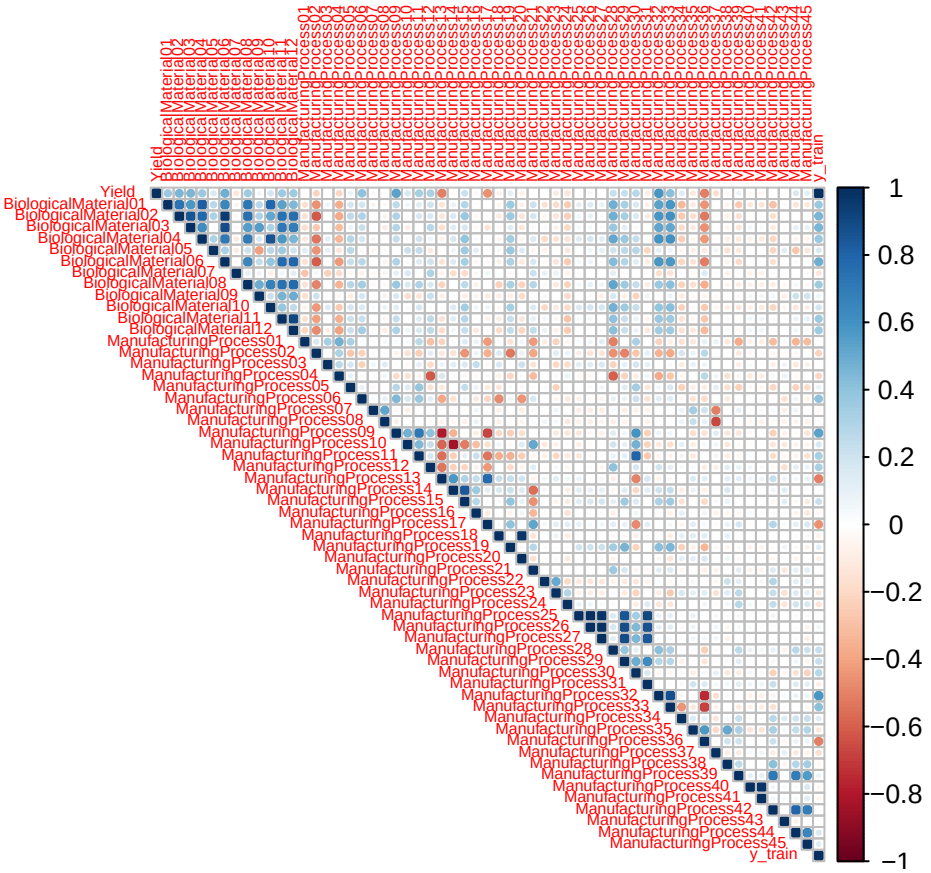
```
set.seed(123)

split_index <- createDataPartition(data_imputed_2$Yield, p = 0.7, list = FALSE)
X_train <- data_imputed_2[split_index, ]
y_train <- data_imputed_2$Yield[split_index]

X_test <- data_imputed_2[-split_index, ]
y_test <- data_imputed_2$Yield[-split_index]
```

```
dim(X_train)
```

```
## [1] 124  58
```

Removing pairwise correlated values.

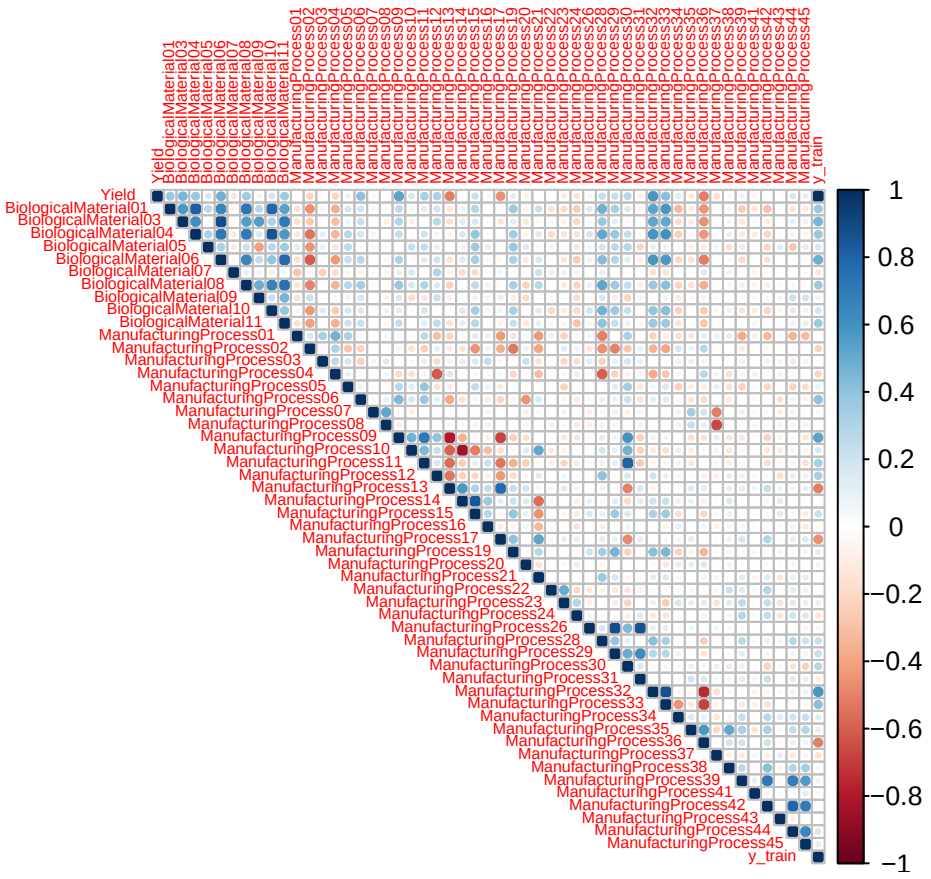
```
correlations  <- cor(cbind(X_train, y_train), use = "pairwise.complete.obs")
corrplot::corrplot(correlations , type = "upper", tl.cex = 0.5, mar = c(0, 0.2, 0, 0))
```



```
correlations_threshold  <- 0.9
highCorr <- findCorrelation(cor(X_train), correlations_threshold)
correlations_threshold_Pred <- names(X_train)[highCorr]
X_train_regression <- X_train[, -highCorr]
X_test_regression <- X_test[, -highCorr]
dim(X_train_regression)
```

```
## [1] 124  52
```

```
correlations_regression <- cor(cbind(X_train_regression, y_train), use = "pairwise.complete.obs")
corrplot::corrplot(correlations_regression, type = "upper", tl.cex = 0.5, mar = c(0, 0.2, 0, 0))
```

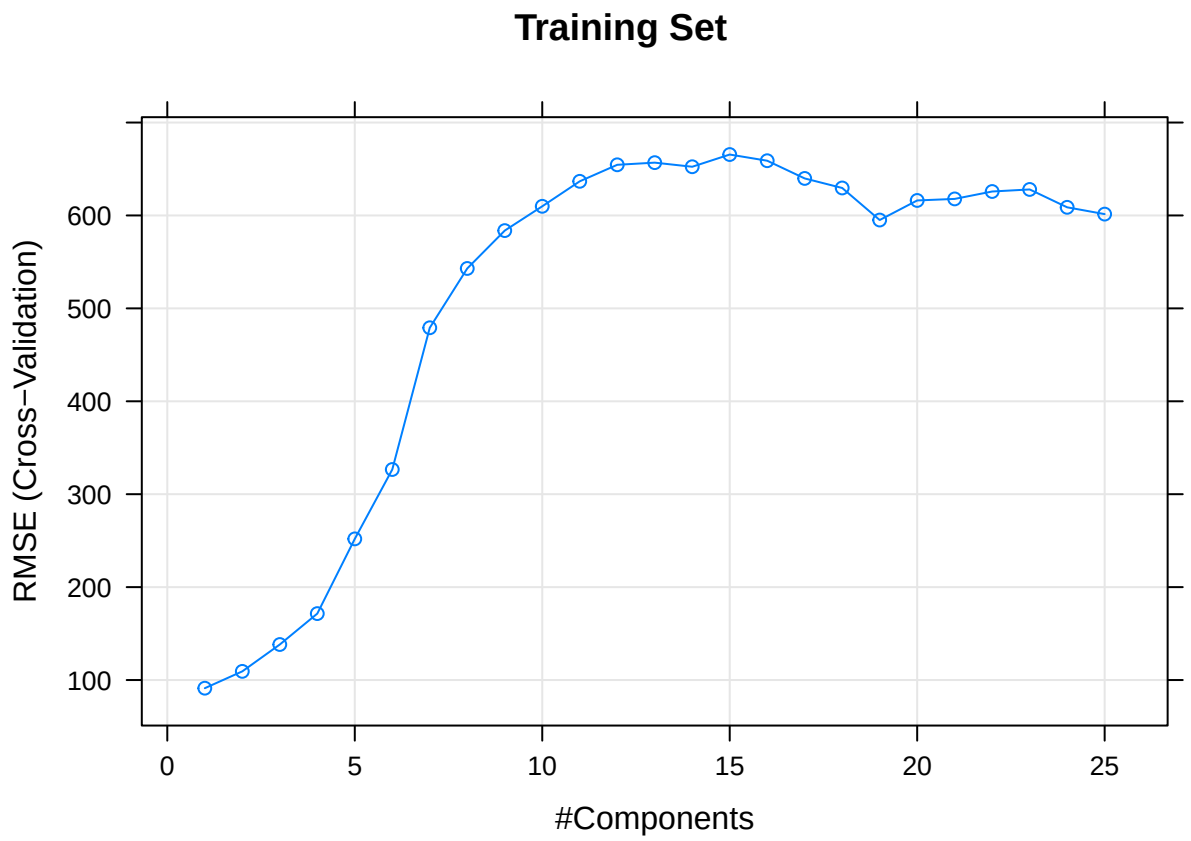



```
set.seed(123)

control <- trainControl(method = "cv", number = 10)

lmFit <- train(x = X_train_regression, y = y_train, method = "lm", trControl = control)
lmFit
```

```
## Linear Regression
##
## 124 samples
##  52 predictor
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 110, 112, 112, 112, 112, 112, ...
## Resampling results:
##
##   RMSE          Rsquared  MAE
##   1.332945e-13  1         4.603432e-14
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
set.seed(123)
pls_model <- train(Yield ~ ., data = X_train_regression, method = "pls", center = TRUE, scale = TRUE, t
plot(pls_model,  val.type = "RMSEP", main = "Training Set")
```




```
pls_model$results %>% filter(ncomp == pls_model$bestTune$ncomp) %>% select(ncomp, RMSE, Rsquared)
```

```
##   ncomp     RMSE  Rsquared
## 1     1 91.23791 0.3863318
```

(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?

```
lmFit1 <- lm(y_train ~ ., cbind(X_train_regression, y_train))
lmPred1 <- predict(lmFit1, X_test_regression)
head(lmPred1)
```

```
##     1     2     3     4     5    10
## 38.00 42.44 42.03 41.42 42.49 42.45
```

```
lmValues1 <- data.frame(obs = y_test, pred = lmPred1)
defaultSummary(lmValues1)
```

```
##         RMSE     Rsquared          MAE
## 1.481299e-14 1.000000e+00 1.257114e-14
```

```
pls_prediction <- predict(pls_model, newdata = X_test)

results <- data.frame(Model = "PLS", RMSE = caret::RMSE(pls_prediction, y_test), Rsquared = caret::R2(pl
results
```

```
##   Model     RMSE  Rsquared
## 1   PLS 1.220048 0.5137327
```

I did two models. But, the RMSE of PLS model is lower. So, I would opt for this.

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

```
pls_importance <- varImp(pls_model)$importance %>% as.data.frame() %>% rownames_to_column("Variable") %
```

```
## Warning: package 'pls' was built under R version 3.6.3
```

```
##
## Attaching package: 'pls'
```

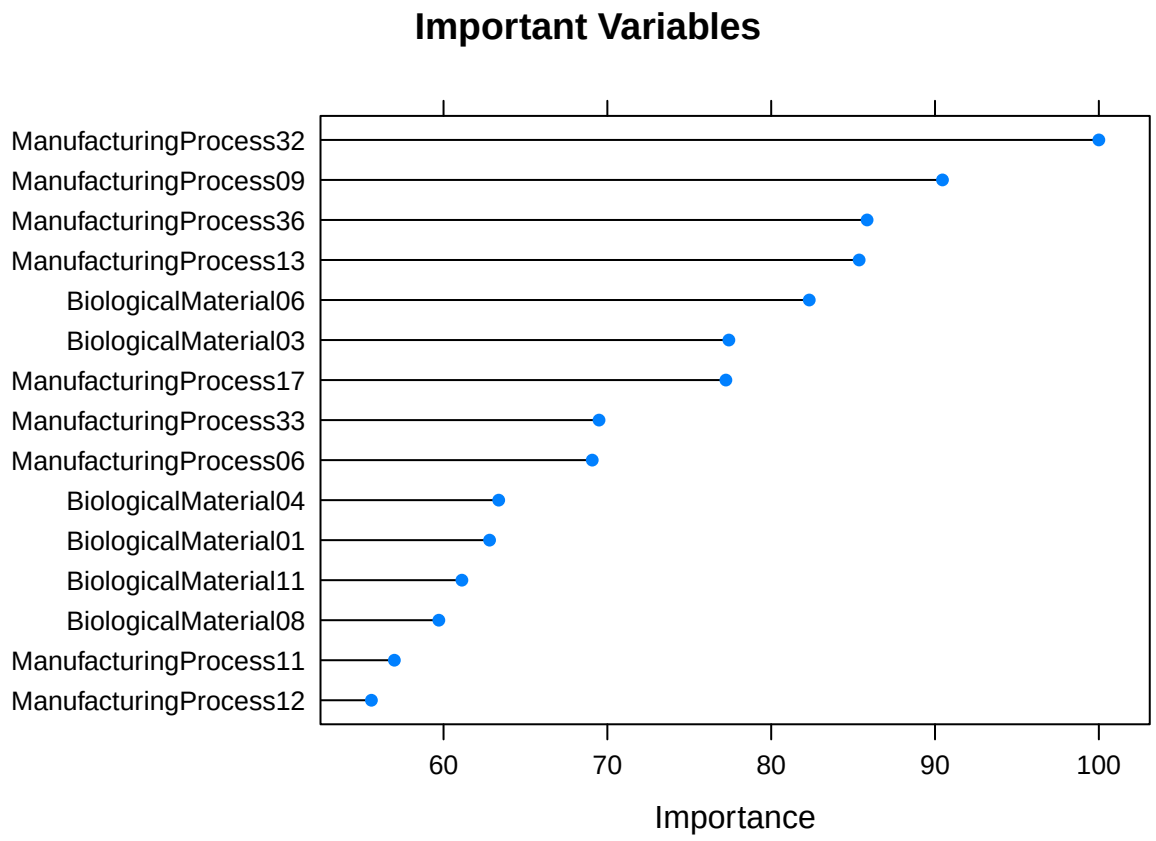
```
## The following object is masked from 'package:fpp2':
##
##     gasoline
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
varImp(pls_model) %>% plot(., top = max(pls_importance$importance), main = "Important Variables")
```




```
pls_importance %>% mutate(Variable = gsub("[0-9]+", "", Variable)) %>% group_by(Variable) %>% tally() %
```

```
## # A tibble: 2 x 2
##   Variable                n
##   <chr>               <int>
## 1 ManufacturingProcess     9
## 2 BiologicalMaterial       6
```

There are 15 important variables, of which 9 are from ManufacturingProcess and 6 from BiologicalMaterial. So, the process predictors dominate the list.

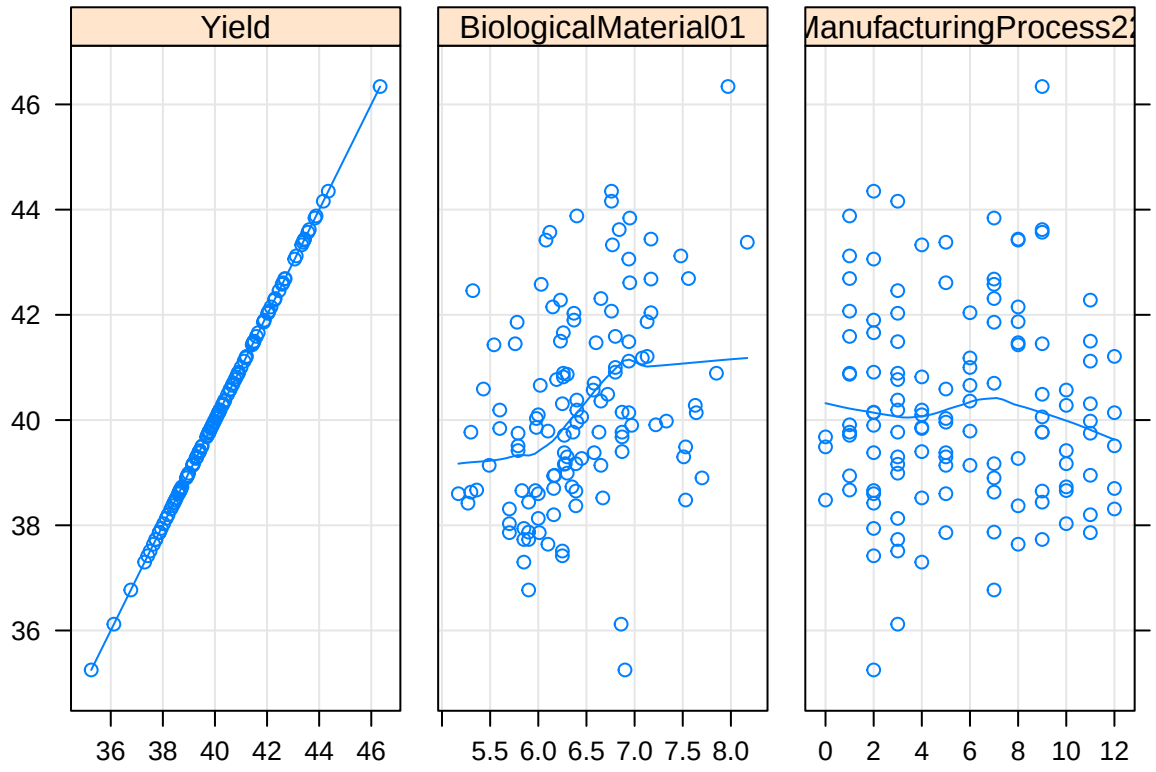
(f) Explore the relationships between each of the top predictors and the response. How could this information be helpful in improving yield in future runs of the manufacturing process?

The corrplot for the acted up, so I used the lm model for this one.

```
lmImp <- varImp(lmFit, scale = FALSE)
```

```
## Warning in summary.lm(object): essentially perfect fit: summary may be
## unreliable
```

```r
viporder <- order(abs(lmImp$importance), decreasing=TRUE)

topVIP = rownames(lmImp$importance)[viporder[c(1:3)]]

featurePlot(X_train_regression[, topVIP], y_train, plot = "scatter", between = list(x = 1, y = 1), type
```




BiologicalMaterial01 and ManufacturingProcess22 appear to be seem to be important and the yield is a linear graph.