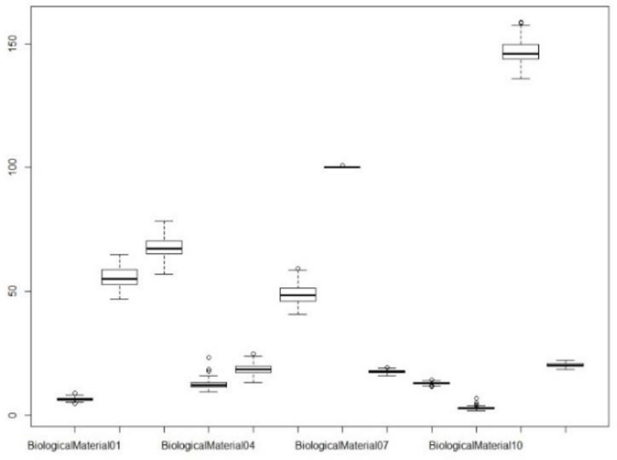
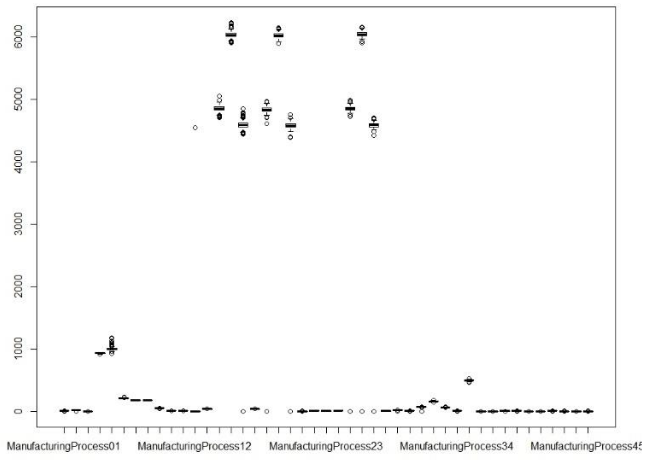
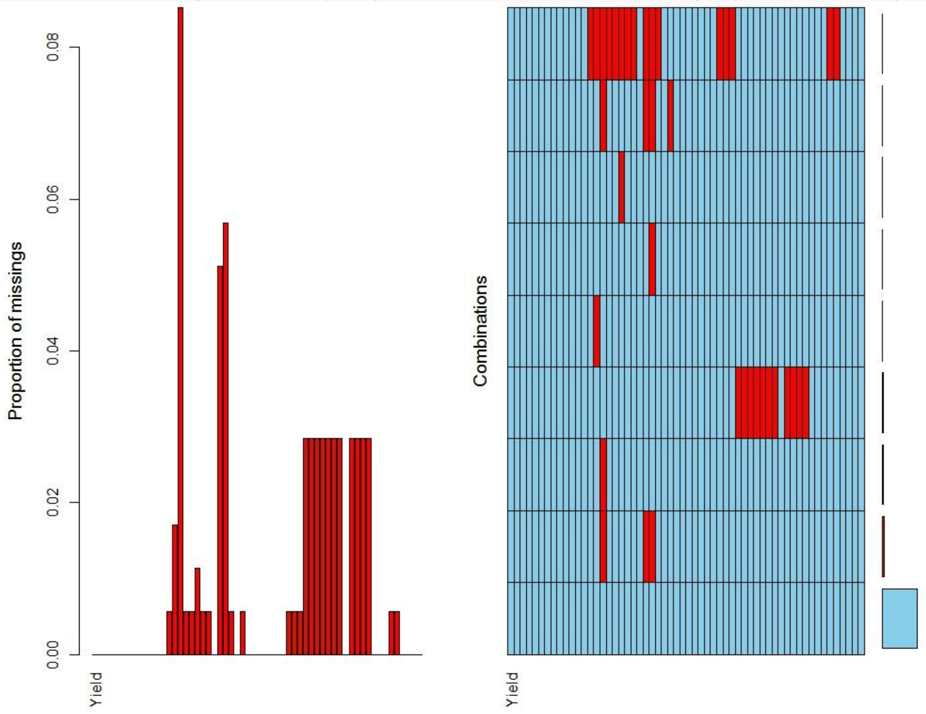
```r
#================================================================================
#                         Using R: Chemical manufacturing process
#================================================================================
#The objective is to understand the relationship between biological 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: This data set contains information about a chemical manufacturing process, in
which the goal is to understand the relationship between the process and the resulting
final product yield. Raw material in this process is put through a sequence of 27 steps
to make the final pharmaceutical product. The starting material is generated from a
biological unit and has a range of quality and characteristics. The objective in this
project was to develop a model to predict percent yield of the manufacturing process. The
data set consisted of 177 samples of biological material for which 57 characteristics
were measured. Of the 57 characteristics, there were 12 measurements of the biological
starting material and 45 measurements of the manufacturing process. The process variables
included measurements such as temperature, drying time, washing time, and concentrations
of by-products at various steps. Some of the process measurements can be controlled,
while others are observed. Predictors are continuous, count, categorical; some are
correlated, and some contain missing values. Samples are not independent because sets of
samples come from the same batch of biological starting material.
#--------------------------------------------------------------------------------
#(a) loading 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.
boxplot(ChemicalManufacturingProcess[,1])
boxplot(ChemicalManufacturingProcess[,2:13])
boxplot(ChemicalManufacturingProcess[,14:58])

#The box plot of the data for biological (on the left) and for the processing data (on
right) has shown below. As can be seen almost all of the predictors have consistent
range, though their values are different. So, we skip the centering and scaling of the
data. Although they could be useful, centering and scaling highly recommend for data with
significant different range.
```

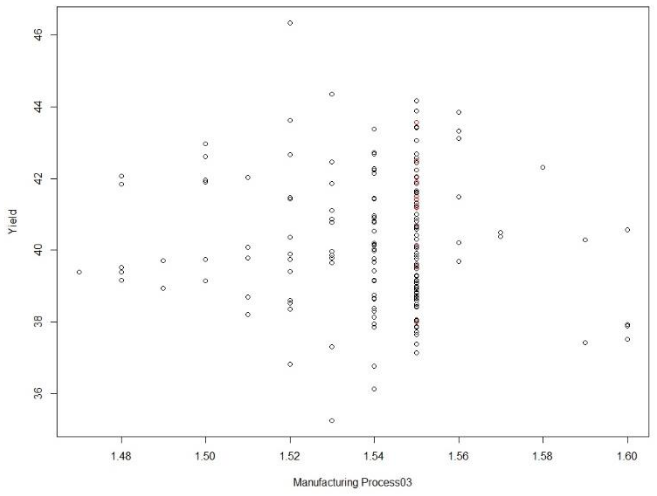


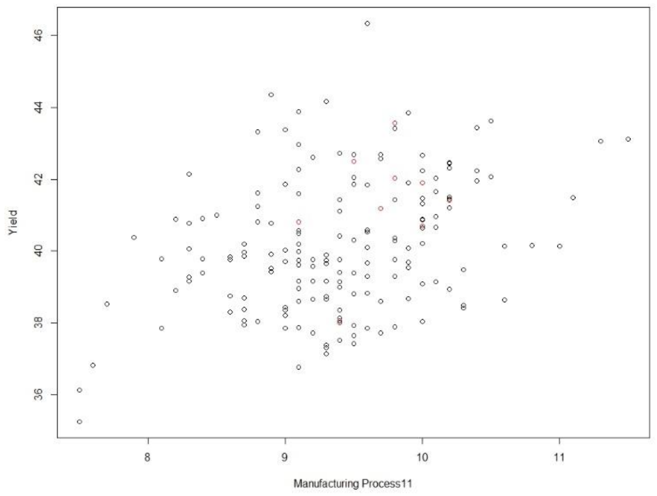


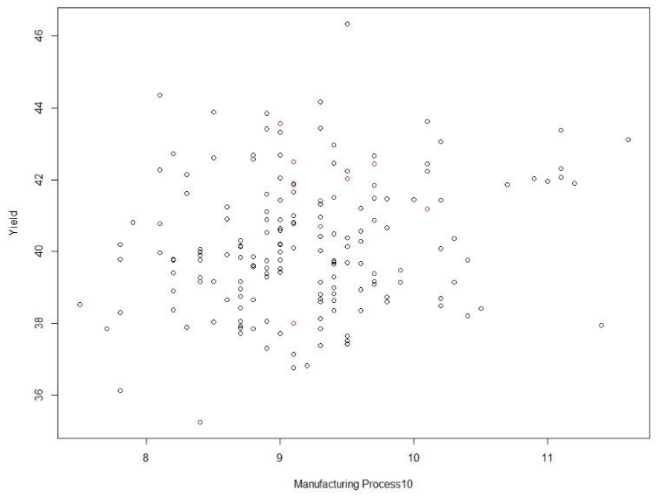
```
#-----------------------------------------------------------------------------------
#(b) A small percentage of cells in the predictor set contain missing values. We use an imputation function to fill in these missing values.

#let's explore data
head(ChemicalManufacturingProcess)
summary(ChemicalManufacturingProcess)
sum(is.na(ChemicalManufacturingProcess))
#There are 106 variables missing. It is small enough (less than 10 percent) to be imputed reasonably. There is no missing in biological and yield. All the missing is in process. Also, the percentage of missing for each variable and total is less than 10 percent.

#Missing pattern
library(VIM)
summary(aggr(ChemicalManufacturingProcess))
The missing pattern and the frequency of them are shown in figure at right hand side.
```



```
#There are many methods and some of them are robust. However, we use the method of K nearest neighbor to impute the missing data in this project. In theory people recommend choosing large number for K to avoid noise. However, there is no guarantee that it would be better. Cross-validation is another method to optimize K. Usually k between 3 to 10 is recommended as a rule of thumb. For this project we use 9 which is an odd number recommended.

set.seed(100)
temp <- kNN(ChemicalManufacturingProcess, k=9)
imputed.data <- temp[,1:58]
sum(is.na(imputed.data))
summary(aggr(imputed.data))

#Some imputed can be shown
plot(temp$ManufacturingProcess03, temp$Yield,
     col=factor(temp$ManufacturingProcess03_imp), xlab="Manufacturing Process03",
ylab="Yield")
plot(temp$ManufacturingProcess11, temp$Yield,
     col=factor(temp$ManufacturingProcess11_imp), xlab="Manufacturing Process11",
ylab="Yield")
plot(temp$ManufacturingProcess10, temp$Yield,
     col=factor(temp$ManufacturingProcess10_imp), xlab="Manufacturing Process10",
ylab="Yield")

#Some of the imputed data can be seen in the following plots selected by random.
```



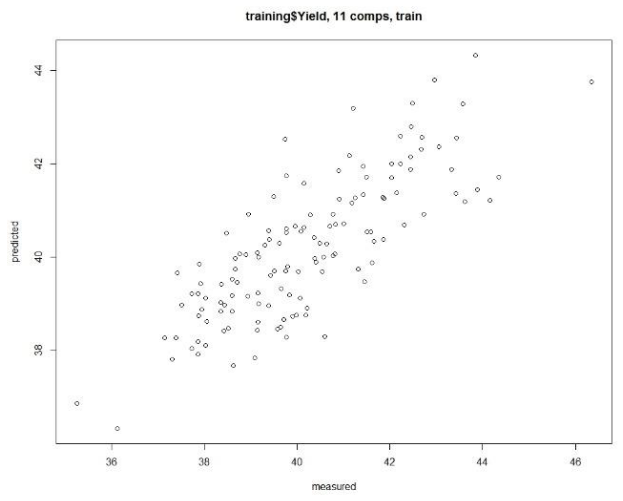


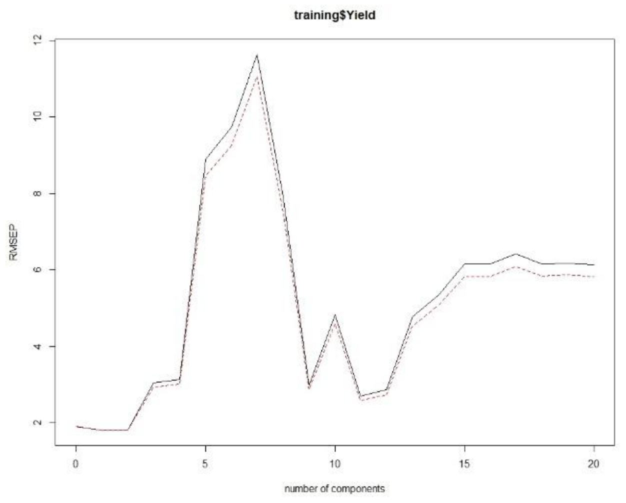


```
#-------------------------------------------------------------------------------------
#(c) Splitting the data into a training and a test set, pre-processing the data, and
tuning hyper parameters.

library(caret)
nzv <- nearZeroVar(imputed.data, saveMetrics= TRUE)
nzv[nzv$nzv,]
# Using nearZeroVar function, there is one predictor that has a near zero variance, but
it belongs to Biological Materia. So, we ignore it.

#Linear Dependencies
```

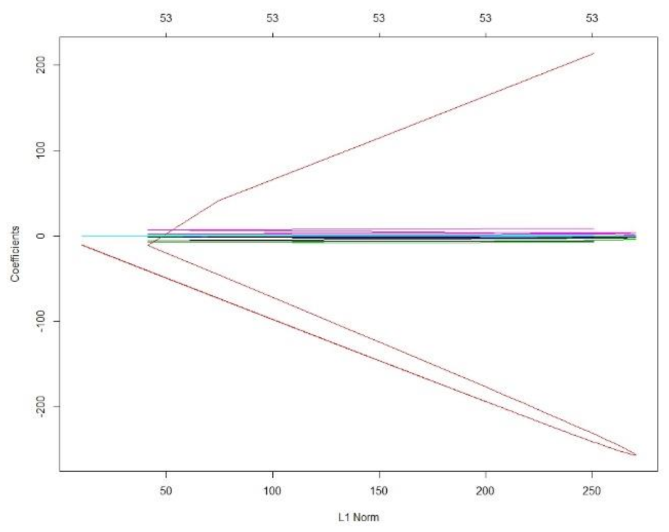
```r
comboInfo <- findLinearCombos(imputed.data)
imputed.data <- imputed.data[, -comboInfo$remove]
dim(imputed.data)
#This function has recognized one combination that has linear relationship. Here we
remove this predictor which belongs to the process data.

#Assessing Correlation
correlation <- round(cor(imputed.data), 3)
highCorr <- sum(abs(correlation[upper.tri(correlation)]) > 0.99)
summary(highCorr)
highlyCorDescr <- findCorrelation(correlation, cutoff=0.99)
imputed.data <- imputed.data[,-highlyCorDescr]
descrCor2 <- cor(imputed.data)
summary(descrCor2[upper.tri(descrCor2)])
dim(imputed.data)
#There are 3 predictors which are highly correlated (|correlation| > 0.99). We remove
these columns. Now, there is 53 predictors left.

#Now, we use Simple Splitting. As long as we do not need validating set, we split data in
to two parts of training and test. We assign the weight for training in the way that we
have enough observation and they be more than the number of predictors. So, about 75% for
training and 25% for test would be used here.

#Spliting Data
set.seed(100)
tra <- createDataPartition(imputed.data[,1], p=.75, list=FALSE, times=1)
head(tra)
training <- data.frame(imputed.data[tra,])
test <- data.frame(imputed.data[-tra,])

#Now, we try three different methods of PLS, Ridge and LASSO Regression.

#Partial Least Squares
library(pls)
fit.pls <- plsr(training$Yield ~ ., 20, data=training, method="oscorespls",
validation="CV")
summary(fit.pls)
dev.off()
plot(RMSEP(fit.pls))

#It seems using only 2 components is enough. However, I used 11 component (the next
minimum) which gives better fit.
ncom <- 11
fit.pls <- plsr(training$Yield ~ ., ncom, data=training, method="oscorespls")
plot(fit.pls)
summary(fit.pls)

#R2 (R Square)
SStot <- sum((training$Yield-mean(training$Yield))^2)
SSres <- sum((training$Yield-drop(predict(fit.pls, training[,2:dim(training)[2]],
ncom)))^2)
Rsqu.pls <- 1-(SSres/SStot)
resid.pls <- drop(fit.pls$resid)[,ncom]
MSE.pls <- mean(resid.pls^2)
```

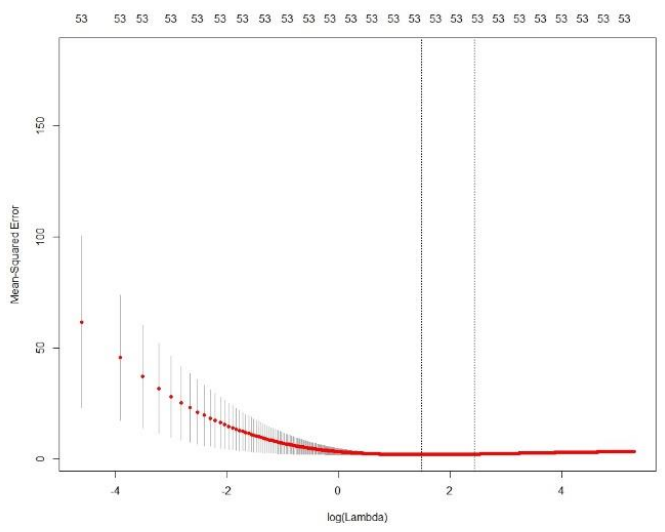
```
#The metrics are MSE=1.269 and R2=0.642
```



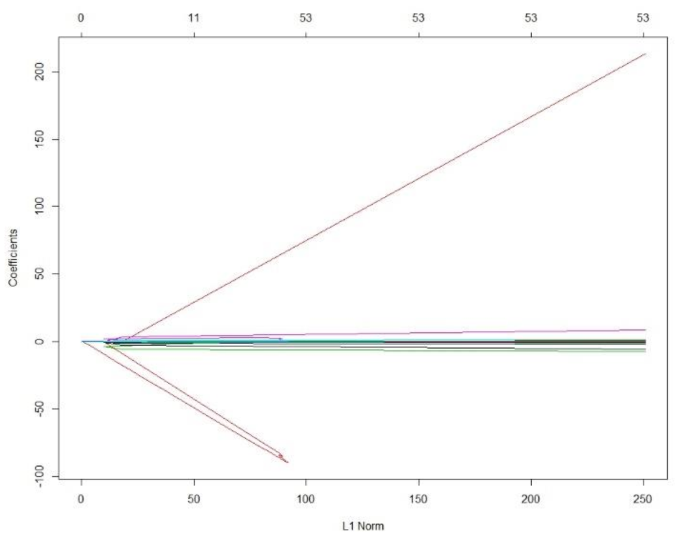


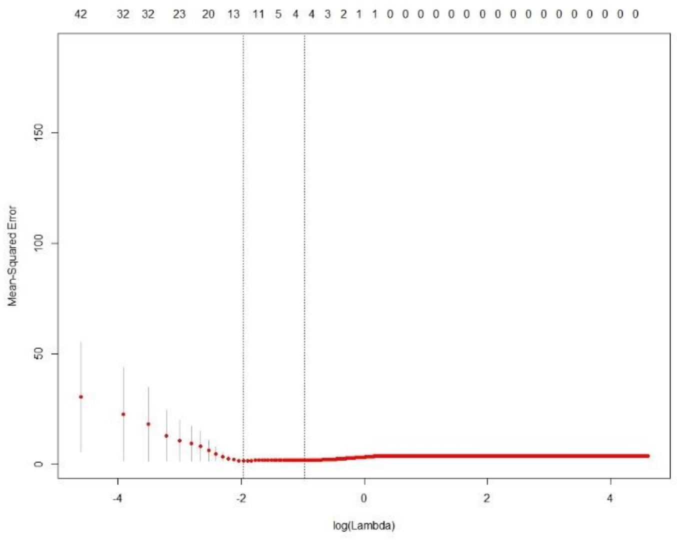
```
#Ridge Regression
#we use glmnet with alpha equal to 0 to perform ridge regression
library(glmnet)
fit.ridge <- glmnet(as.matrix(training[,2:dim(training)[2]]), training$Yield, alpha=0,
lambda=seq(0,200,0.01))
plot(fit.ridge)
cv.out <- cv.glmnet(as.matrix(training[,2:dim(training)[2]]), training$Yield, alpha=0,
lambda=seq(0,200,0.01))
plot(cv.out)
lambda.ridge <- cv.out$lambda.min
fit.ridge <- glmnet(as.matrix(training[,2:dim(training)[2]]), training$Yield, alpha=0,
lambda=lambda.ridge)

#R2 (R Square)
SStot <- sum((training$Yield-mean(training$Yield))^2)
SSres <- sum((training$Yield-predict(fit.ridge, s=lambda.ridge,
newx=as.matrix(training[,2:dim(training)[2]])))^2)
Rsqu.ridge <- 1-(SSres/SStot)
resid.ridge <- training$Yield-predict(fit.ridge, s=lambda.ridge,
newx=as.matrix(training[,2:dim(training)[2]]))
MSE.ridge <- mean(resid.ridge^2)
#Ridge Regression shrinkage coefficient and tuning graph is shown below. The metrics are
MSE=1.518 and R2=0.572.
```





```
#LASSO Regression by help of glmnet with alpha equal to 1
fit.lasso <- glmnet(as.matrix(training[,2:dim(training)[2]]), training$Yield, alpha=1,
lambda=seq(0,100,0.01))
plot(fit.lasso)
cv.out <- cv.glmnet(as.matrix(training[,2:dim(training)[2]]), training$Yield, alpha=1,
lambda=seq(0,100,0.01))
plot(cv.out)
lambda.lasso <- cv.out$lambda.min
fit.lasso <- glmnet(as.matrix(training[,2:dim(training)[2]]), training$Yield, alpha=1,
lambda=lambda.lasso)
resid.lasso <- training$Yield - predict(fit.lasso, s=lambda.lasso,
newx=as.matrix(training[,2:dim(training)[2]]))
MSE.lasso <- mean(resid.lasso^2)

#R2 (R Square)
SStot <- sum((training$Yield-mean(training$Yield))^2)
SSres <- sum((training$Yield-predict(fit.lasso, s=lambda.lasso,
newx=as.matrix(training[,2:dim(training)[2]])))^2)
Rsqu.lasso <- 1-(SSres/SStot)
# LASSO Regression shrinkage coefficient and tuning parameter are shown below. The
metrics are MSE=1.219 and R2=0.656.
```





```
#-------------------------------------------------------------------------------------
#(d) Predicting the response for the test set.

#Partial Least Squares
test.pls <- drop(predict(fit.pls, test[,2:dim(training)[2]], ncom))
#R2 (R Square)
SStot <- sum((test$Yield-mean(test$Yield))^2)
SSres <- sum((test$Yield-test.pls)^2)
Rsqu.pls.test <- 1-(SSres/SStot)
resid.pls <- test$Yield-test.pls
test.MSE.pls <- mean(resid.pls^2)

#Ridge Regression
test.ridge <- predict(fit.ridge, s=lambda.ridge
,newx=as.matrix(test[,2:dim(training)[2]]))
#R2 (R Square)
SStot <- sum((test$Yield-mean(test$Yield))^2)
SSres <- sum((test$Yield-test.ridge)^2)
Rsqu.ridge.test <- 1-(SSres/SStot)
resid.ridge <- test$Yield-test.ridge
test.MSE.ridge <- mean(resid.ridge^2)

#LASSO Regression by help of glmnet with alpha equal to 1
test.lasso <- predict(fit.lasso, s=lambda.lasso ,
newx=as.matrix(test[,2:dim(training)[2]]))
#R2 (R Square)
SStot <- sum((test$Yield-mean(test$Yield))^2)
SSres <- sum((test$Yield-test.lasso)^2)
Rsqu.lasso.test <- 1-(SSres/SStot)
resid.lasso <- test$Yield-test.lasso
test.MSE.lasso <- mean(resid.lasso^2)

#Here in this problem the R-Squared are used as a performance metric. By comparing these
assessed methods, the LASSO has been chosen.
```

Method	MSE(Training)	MSE(Test)	R^2 (Training)	R^2 (Test)
PLS	1.269	1.218	0.642	0.580
Ridge	1.518	1.375	0.572	0.526
LASSO	1.219	1.249	0.656	0.570

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

#To identify which predictor is the most important, there are many controversies in the
literature.
#People argued the use of the magnitude of its regression coefficient, but it depends on
the scale of measurements. Others argued about the use of significance level (p-value).
However, the distinction between statistical and practical importance applies here, too.
Even if the predictors are measured on the same scale, a small coefficient that can be
estimated precisely will have a small p-value, while a large coefficient that is not
estimate precisely will have a large p-value.
#To overcome these, standardized regression coefficients are introduced. However, this is
illusory.
```

```
#there is no reason why a change of one SD in one predictor should be equivalent to a
change of one SD in another predictor. Some variables are easy to change, others are more
difficult. The answer to which variable is most important depends on the specific context
and why the question is being asked.
#In this case, I ignored because of the lack of information about the predictors, and I
assumed all of them are in the same range and scale and the measurement are consistent
and they have the same likelihood of changing. So, I tried to see the ratio of the
participation of each predictor in the response.

B0 <- fit.lasso$a0
B <- as.matrix(fit.lasso$beta)
B[B==0] <- NA
View(B)
#As can be seen B03, B05, MP06, MP09, MP13, MP15, MP17, MP32, MP36, MP37, MP39 and MP45
are used to fit regression in LASSO and the others are 0.
X <- imputed.data[,2:dim(imputed.data)[2]]
X <- X[,is.na(B)==F]
head(X)
Pre <- X
B <- B[is.na(B)==F]
for (i in 1:length(B)){
  Pre[,i] <- Pre[,i]*B[i]
}
Res <- matrix(B0, nrow=dim(Pre)[1], ncol=1)
for(i in 1:dim(Pre)[1]){
  for(j in 1:dim(Pre)[2]){
    Res[i] <- Res[i] + Pre[i,j]
  }
}
#Calculating each ratio
Ratio <- matrix(0, nrow=dim(Pre)[1], ncol=dim(Pre)[2])
for(i in 1:dim(Pre)[1]){
  for(j in 1:dim(Pre)[2]){
    Ratio[i,j] <- abs((Pre[i,j])/(Res[i]-B0))
  }
}
#finding maximum ratio by max
Max.rat <- matrix(0, nrow=dim(Pre)[2], ncol=1)
for(i in 1:dim(Pre)[2]){
  Max.rat[i] <- max(Ratio[,i])
}
#finding maximum ratio by mean
Max.rat.mean <- matrix(0, nrow=dim(Pre)[2], ncol=1)
for(i in 1:dim(Pre)[2]){
  Max.rat.mean[i] <- mean(Ratio[,i])
}
# At first, I used the coefficient to find the value of each predictor and by summation
of them I extract the response and then rank each of the predictors in the way that they
have more absolute influence on the response. As can be seen the order of predictors from
the most influence on the lowest one is as follow, by use of average ratio of
observations:
#MP32 MP09  MP17  MP06  MP13  MP36  MP15  MP45  B05   MP39  B03   MP37
```

```r
#As the ranking shows, the process predictors have more influence than biological.
Actually, the last 3 predictors have less than 1 percent influence in the response and
can be ignored. If we use the max of ratio we reach to the same ranking as follow:
#MP32 MP09  MP17  MP06  MP13  MP15  MP36  MP05  B45   MP39  B03   MP37

#The influential rate can be seen in the table below:
By use of average between values of each predictor (in percentile):
```

MP32	MP09	MP17	MP06	MP13	MP15	MP36	B05	MP45	MP39	B03	MP37
45.67	25.52	12.78	5.19	2.99	2.44	2.41	1.19	1.17	0.29	0.19	0.16

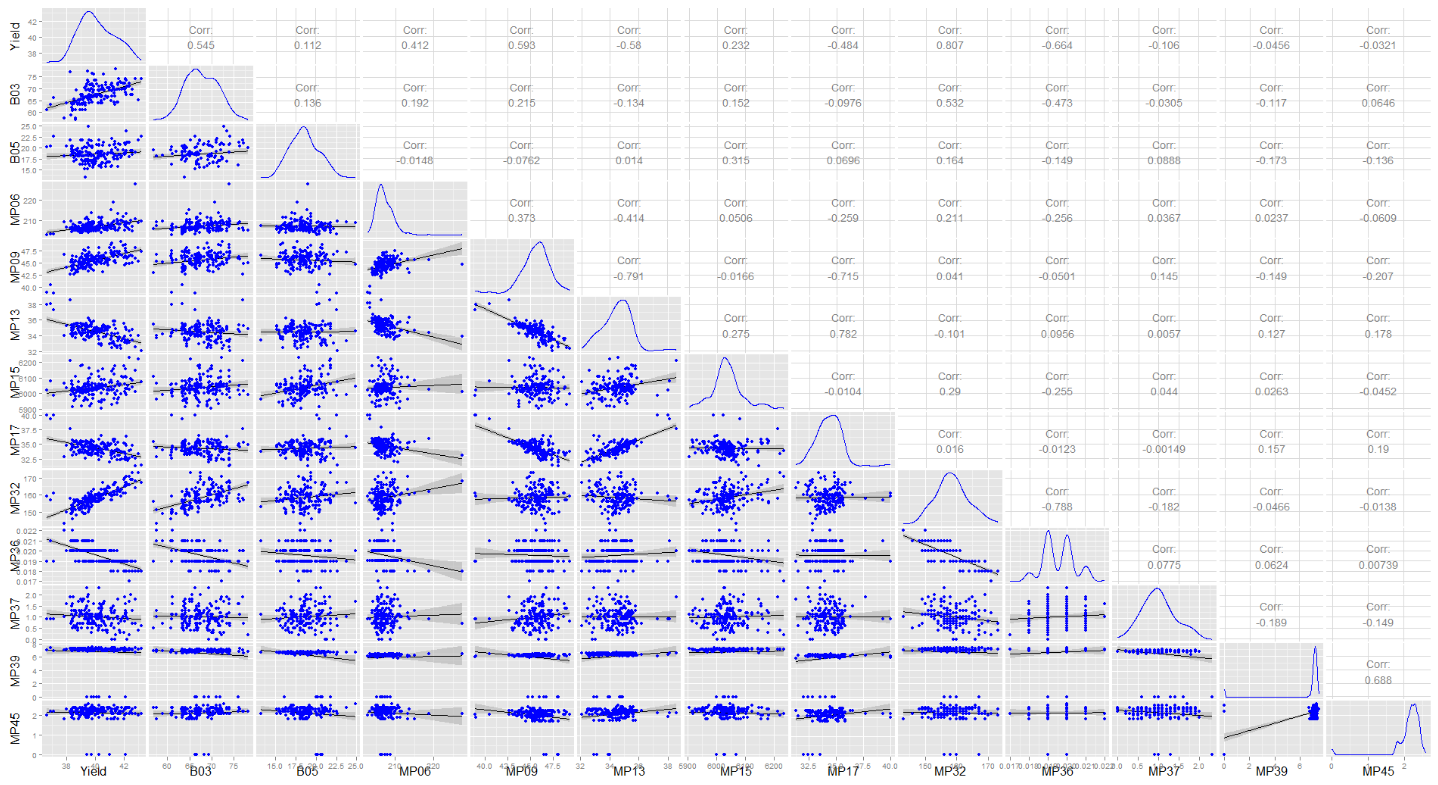
```r
# By use of maximum between values of each predictor (in percentile):
```

MP32	MP09	MP17	MP06	MP13	MP15	MP36	MP45	B05	MP39	B03	MP37
46.69	26.43	11.27	5.45	2.79	2.51	2.28	1.06	0.99	0.27	0.18	0.08

```r
#---------------------------------------------------------------------------------------
#(f) Exploring the relationships between each of the top predictors and the response.

Relationship <- cbind(B03=X[,1], B05=X[,2], MP06=X[,3], MP09=X[,4], MP13=X[,5],
MP15=X[,6], MP17=X[,7], MP32=X[,8], MP36=X[,9], MP37=X[,10], MP39=X[,11], MP45=X[,12])
Relationship <- cbind(Yield=cbind(Yield=Res, X)[,1], Relationship)

#Assessing Correlation
corre <- abs(round(cor(Relationship),3))
#The ranking of the correlation from high to low is as follow:
```

MP32	MP36	MP09	MP13	B03	MP17	MP06	MP15	B05	MP37	MP39	MP45
0.807	0.664	0.593	0.58	0.545	0.484	0.412	0.232	0.112	0.106	0.046	0.032

```r
library(GGally)
ggpairs(Relationship, lower=list(continuous="smooth", params=c(colour="blue")),
        diag=list(params=c(colour="blue")), upper=list(params=list(corSize=6)),
axisLabels='show')
```



```
#As can be seen some predictors have both high influence and correlation such as MP32,
MP09, MP17. Some of them have intermediate influence and correlation such as MP06, MP13
and MP15. Other parameter like MP36 has high correlation but it has very low influence on
the response. I think we should put most of our effort to improve these process
predictors, because by spending our money and time we can improve the result more. Or
even improve our modeling by paying special attention to these predictors.
```