DATA624-HW7-Linear-Regression

Kuhn-Johnson exercises 6.2, 6.3

Michael Y.

4/5/2020

Contents

Homework 7 - Linear Regression

In Kuhn and Johnson do problems 6.2 and 6.3. There are only two but they consist of many parts.

Please submit a link to your Rpubs and submit the .rmd file as well.

```r
knitr::opts_chunk$set(echo = TRUE)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(kableExtra)
library(AppliedPredictiveModeling)
library(moments)
library(pls)
```

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

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

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

```r
library(VIM)
```

```
## Loading required package: colorspace

## Loading required package: grid

## Loading required package: data.table

## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##          Please use the package to use the new (and old) GUI.

## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```r
library(mice)
```

```
##
## Attaching package: 'mice'

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
library(olsrr)
```

```
##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded

##
## Attaching package: 'corrplot'

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

```r
options(scipen = 999, digits=7)
```

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.

Permeability Data

Description This pharmaceutical data set was used to develop a model for predicting compounds' **permeability**.

In short, **permeability** is the measure of a molecule's ability to cross a **membrane.**

The body, for example, has notable membranes between the body and brain, known as the **blood-brain barrier**, and between the gut and body in the intestines.

These membranes help the body guard critical regions from receiving undesirable or detrimental substances.

For an orally taken drug to be effective in the brain, it first must pass through the intestinal wall and then must pass through the blood-brain barrier in order to be present for the desired neurological target.

Therefore, a compound's ability to permeate relevant biological membranes is critically important to understand early in the drug discovery process.

Compounds that appear to be effective for a particular disease in research screening experiments, but appear to be poorly permeable may need to be altered in order improve permeability, and thus the compound's ability to reach the desired target.

Identifying permeability problems can help guide chemists towards better molecules.

Permeability assays such as **PAMPA** and **Caco-2** have been developed to help measure compounds' permeability (Kansy et al, 1998).

These screens are effective at quantifying a compound's permeability, but the assay is expensive labor intensive.

Given a sufficient number of compounds that have been screened, we could develop a **predictive model for permeability** in an attempt to potentially reduce the need for the assay.

In this project there were **165 unique compounds**; **1107 molecular fingerprints** were determined for each.

A **molecular fingerprint** is a binary sequence of numbers that represents the **presence or absence** of a specific molecular sub-structure.

The **response** is **highly skewed**,
the **predictors** are **sparse** (15.5 percent are present), and
many predictors are **strongly associated**.

Usage
`data(permeability)`
Value

- `permeability`: permeability values for each compound. (A vector of 165 numbers.)
- `fingerprints`: a 165x1107 matrix of binary fingerprint indicator variables.

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

```r
library(AppliedPredictiveModeling)
data(permeability)
```

- The matrix `fingerprints` contains the 1,107 binary molecular predictors for the 165 compounds, while
- `permeability` contains permeability response.

```r
# dim
dim(permeability)
```

Examine the permeability data

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

```r
N=length(permeability)
# str
str(permeability)
```

```
##  num [1:165, 1] 12.52 1.12 19.41 1.73 1.68 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:165] "1" "2" "3" "4" ...
##   ..$ : chr "permeability"
```

```r
# head
head(permeability)
```

```
##   permeability
## 1       12.520
## 2        1.120
## 3       19.405
## 4        1.730
## 5        1.680
## 6        0.510
```

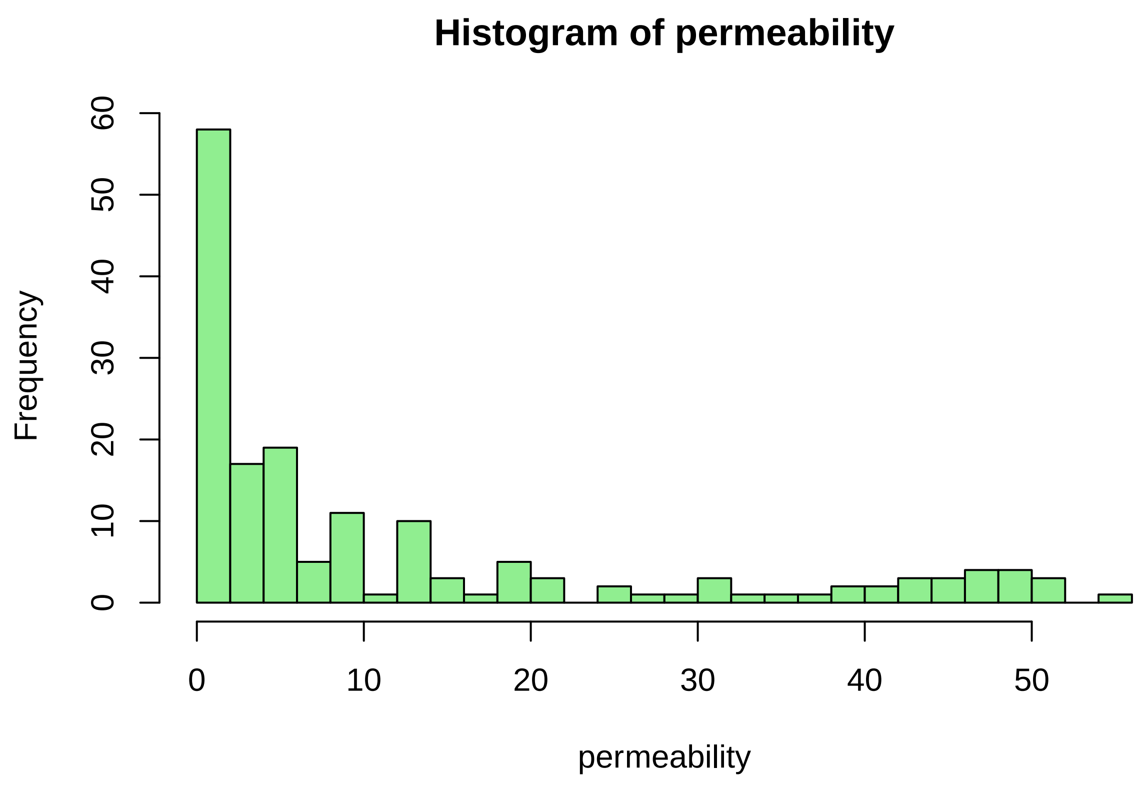
```r
# tail
tail(permeability)
```

```
##     permeability
## 160        0.745
## 161        0.705
## 162        0.525
## 163        1.545
## 164       39.555
## 165        0.795
```

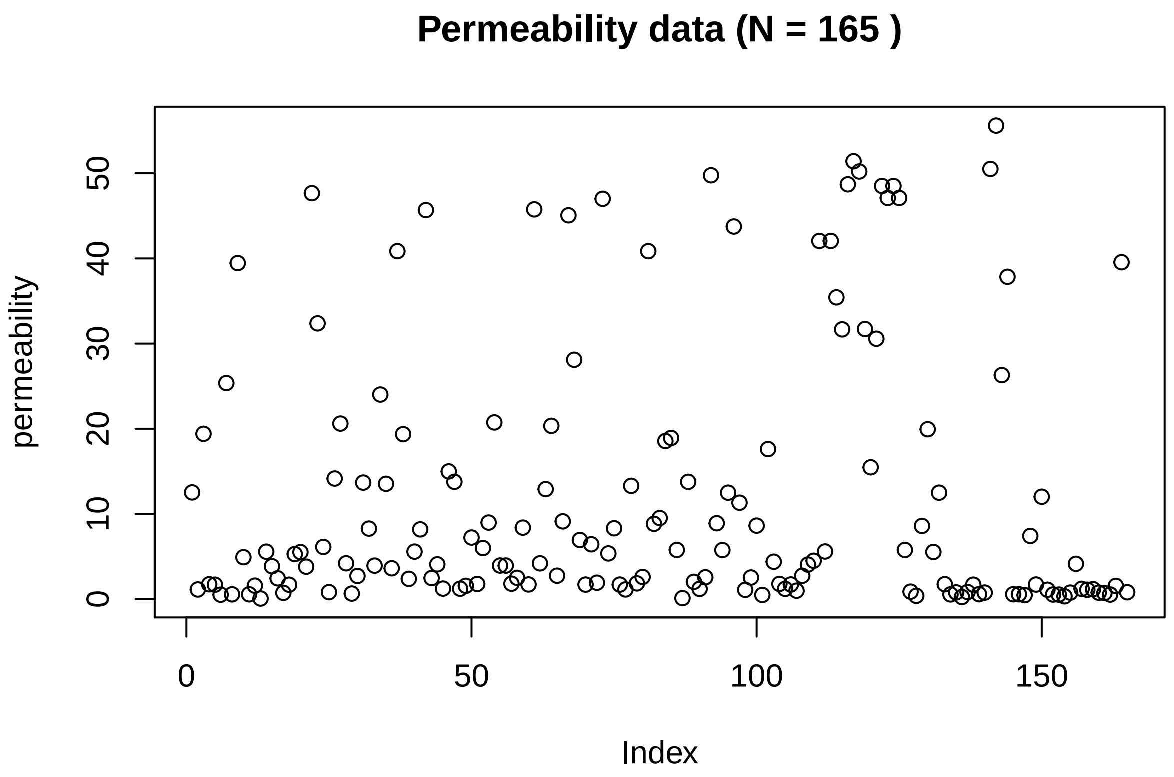
```r
# summary with standard deviation and skewness:
#library(moments)
rbind(summary(permeability),
      paste0("StDev  :",round(sd(permeability),2),"  "),
      paste0("Skew   : ",round(skewness(permeability),2),"  ")) %>%
      as.table()
```

```
##   permeability
##  Min.   : 0.06
##  1st Qu.: 1.55
##  Median : 4.91
##  Mean   :12.24
##  3rd Qu.:15.47
##  Max.   :55.60
##  StDev  :15.58
##  Skew   : 1.41
```

```r
# histogram
hist(permeability,breaks=20,col="lightgreen")
```



```r
# scatterplot
mainlabel=paste("Permeability data (N =",N,")")
plot(permeability,main=mainlabel)
```



The above data is heavily skewed to the right.
Additionally, all of the values for permeability are positive.

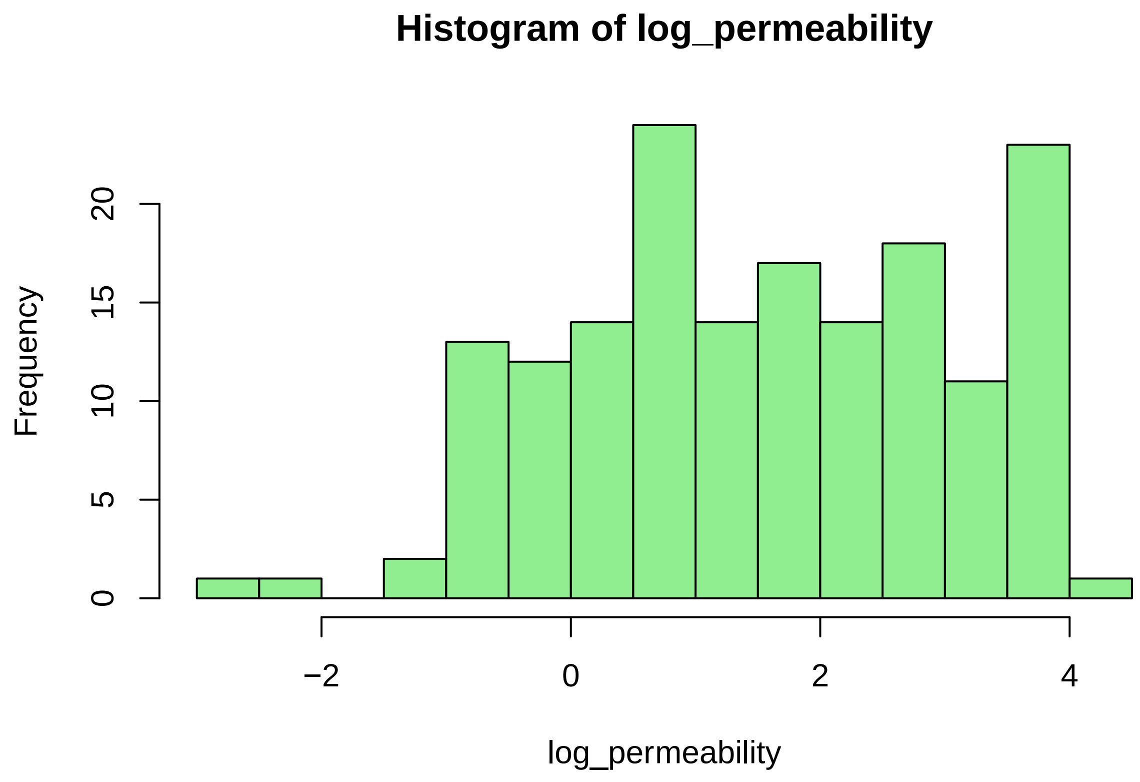
Therefore, we should consider **fitting the log** of the permeability data.
This would ensure that our predicted values are also positive, once we **exponentiate the log results**.

```r
log_permeability = log(permeability)
colnames(log_permeability) <- "log(permeability)"
rbind(summary(log_permeability),
      paste0("StDev  : ",round(sd(log_permeability),2),"  "),
      paste0("Skew   :",round(skewness(log_permeability),2),"  ")) %>%
      as.table()
```

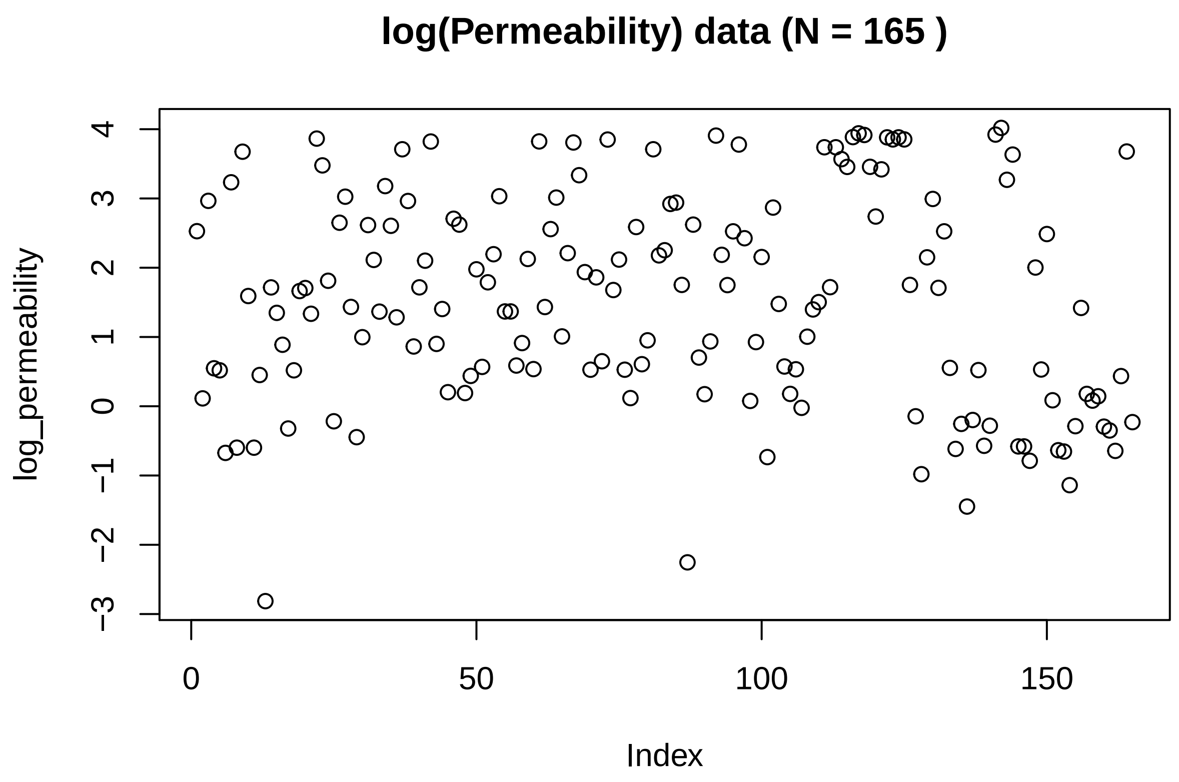
examine the log(permeability)

```
##  log(permeability)
##  Min.   :-2.8134
##  1st Qu.: 0.4383
##  Median : 1.5913
##  Mean   : 1.5464
##  3rd Qu.: 2.7389
##  Max.   : 4.0182
##  StDev  : 1.53
##  Skew   :-0.13
```

```r
# histogram
hist(log_permeability,breaks=20,col="lightgreen")
```



```
# scatterplot
mainlabel=paste("log(Permeability) data (N =",N,")")
plot(log_permeability,main=mainlabel)
```



(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.

```r
fingerprints_nearZeroVarCols <- nearZeroVar( fingerprints)
fingerprints_nTotalCols <- ncol(fingerprints)
fingerprints_nDropCols <- length(fingerprints_nearZeroVarCols)
fingerprints_filtered1 <- fingerprints[,-fingerprints_nearZeroVarCols]
fingerprints_nFiltered <- ncol(fingerprints_filtered1)
dim(fingerprints_filtered1)
```

Filter out the predictors that have low frequencies using the `nearZeroVar` function from the `caret` package.

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

How many predictors are left for modeling? There are 719 columns with nearZeroVar out of 1107, leaving a total of 388 remaining, but there are high correlations between numerous columns.

```r
correl1 <- cor(fingerprints_filtered1)

# determinant is zero -- indicates correlation matrix is singular.
print(paste("Determinant: ", det(correl1)))
```

Check for high correlations between columns in the fingerprints_filtered data set:

```
## [1] "Determinant:  0"
```

```r
# many columns are identical to other columns.
maxcor1 <- max(correl1-diag(1,ncol(correl1),ncol(correl1)))
mincor1 <- min(correl1-diag(1,ncol(correl1),ncol(correl1)))
print(paste("Range of off-diag correlations: ",
            paste0("[",mincor1,",",maxcor1,"]"), "on",ncol(correl1),"columns"))
```

```
## [1] "Range of off-diag correlations:  [-1,1] on 388 columns"
```

```r
# eliminate columns which are identical to other columns
cutoff = 0.999999999
identicals <- findCorrelation(correl1,cutoff = cutoff)
num_identicals <- length(identicals)
print(paste("Quantity of columns which have identical correlation values:",
            num_identicals, "out of",ncol(correl1)))
```

```
## [1] "Quantity of columns which have identical correlation values: 233 out of 388"
```

```r
# drop columns which are identical to some other column
fingerprints_filtered2 <- fingerprints_filtered1[,-identicals]
dim(fingerprints_filtered2)
```

```
## [1] 165 155
```

```r
print(paste("Remaining number of columns: ", ncol(fingerprints_filtered2)))
```

```
## [1] "Remaining number of columns:  155"
```

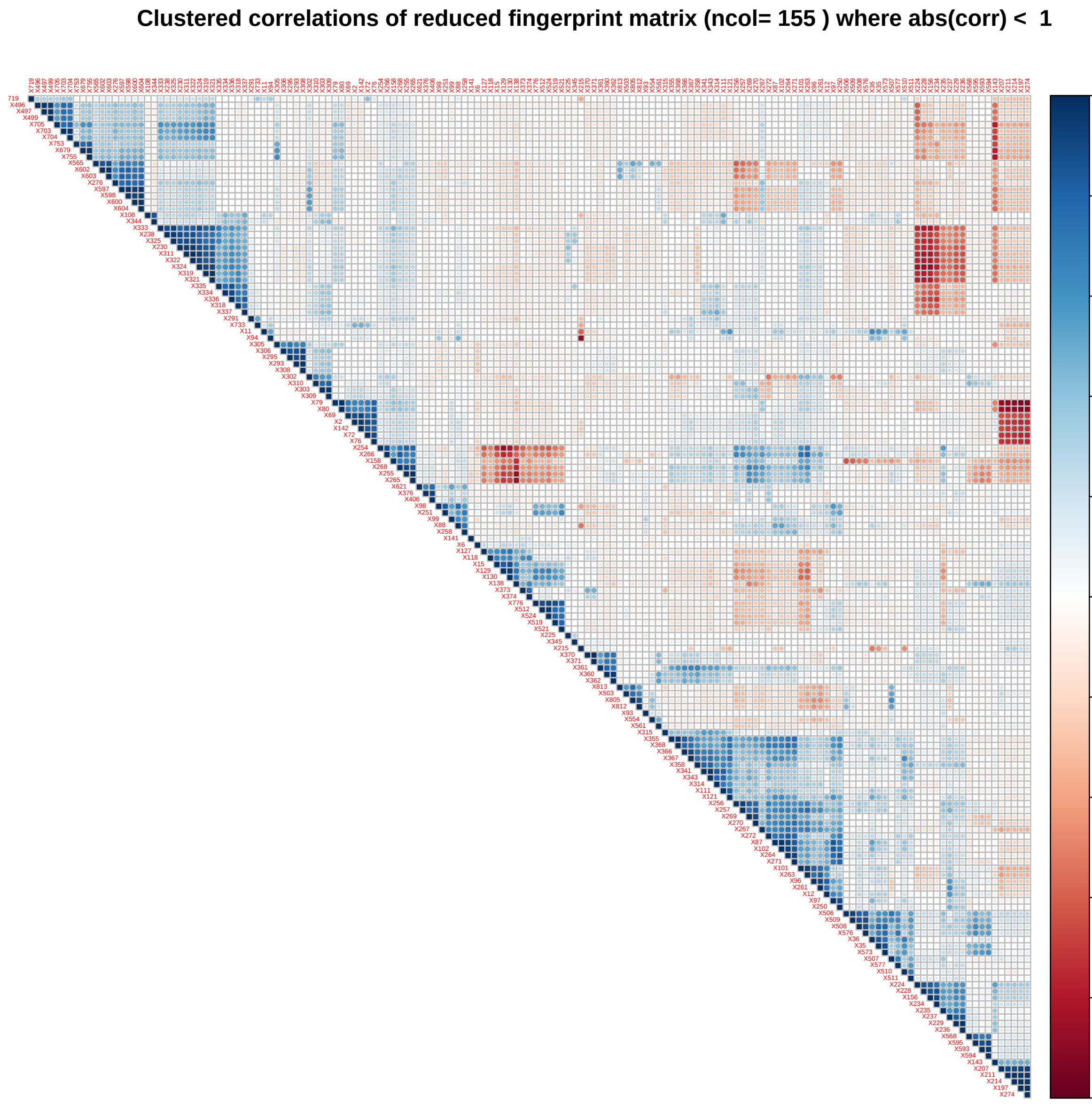
```r
# examine correlations on the reduced matrix
correl2 <- cor(fingerprints_filtered2)
maxcor2 <- round(max(correl2-diag(1,ncol(correl2),ncol(correl2))),5)
mincor2 <- round(min(correl2-diag(1,ncol(correl2),ncol(correl2))),5)
print(paste("Range of off-diag correlations: ",
            paste0("[",mincor2,",",maxcor2,"]"), "on",ncol(correl2),"columns"))
```

```
## [1] "Range of off-diag correlations:  [-0.93315,0.98735] on 155 columns"
```

```r
# determinant is still zero - matrix is singular
print(paste("Determinant: ", det(correl2)))
```

```
## [1] "Determinant:  0"
```

```r
### be sure to specify corrplot::corrplot because the namespace may be masked by pls::corrplot
corrplot::corrplot(
  correl2,
  method = "circle",
  type = "upper",
  order = "hclust",
  tl.cex = 0.3,
  main = paste("\nClustered correlations of reduced fingerprint matrix (ncol=",
               ncol(correl2),") where abs(corr) < ", round(cutoff,4))
)
```



Correlation grid #1

There are still clusters of columns with very high correlations, so let's remove more columns.

```
cutoff = 0.8
high_correl_cols <- findCorrelation(correl1,cutoff = cutoff)
num_high_correl_cols <- length(high_correl_cols)
print(paste("Quantity of columns which have abs(correlation) > ", cutoff ,":",
            num_high_correl_cols, "out of",ncol(correl1)))
```

identify columns with high correlations, and remove

```
## [1] "Quantity of columns which have abs(correlation) >  0.8 : 318 out of 388"
```

```
fingerprints_filtered3 <- fingerprints_filtered1[,-high_correl_cols]
dim(fingerprints_filtered3)
```

```
## [1] 165  70
```

```
print(paste("Remaining number of columns: ", ncol(fingerprints_filtered3)))
```

```
## [1] "Remaining number of columns:  70"
```

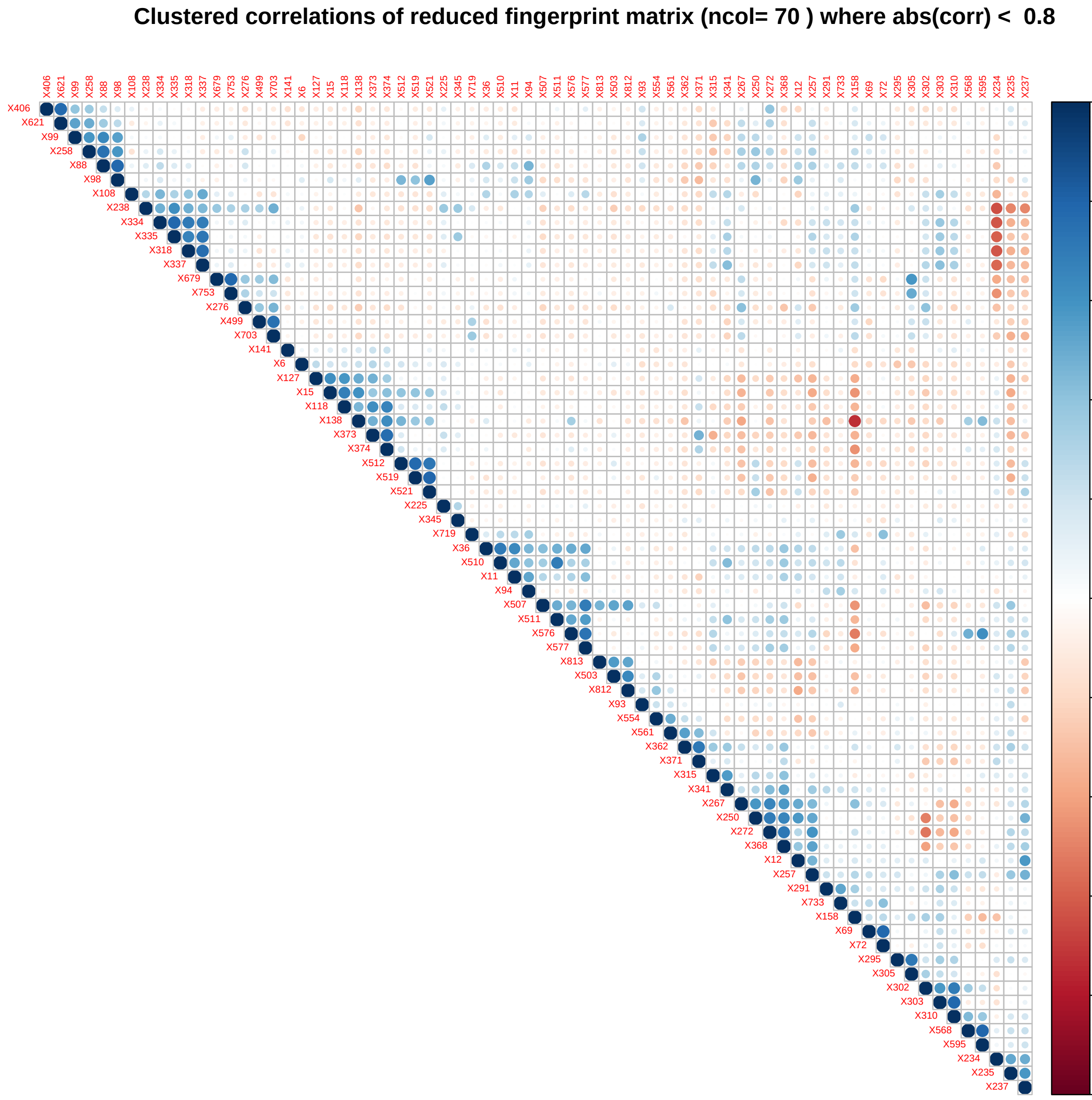
```
correl3 <- cor(fingerprints_filtered3)

maxcor3 <- round(max(correl3-diag(1,ncol(correl3),ncol(correl3))),5)
mincor3 <- round(min(correl3-diag(1,ncol(correl3),ncol(correl3))),5)
print(paste("Range of off-diag correlations: ", paste0("[",mincor3,",",maxcor3,"]"), "on",ncol(correl3)
```

```
## [1] "Range of off-diag correlations:  [-0.73139,0.79436] on 70 columns"
```

```
### Determinant is (barely) nonzero
print(paste("Determinant: ", det(correl3)))
```

```
## [1] "Determinant:  -0.000000000000000000000000000000000000000000000000000000000000000000000000000797
```

```
### be sure to specify corrplot::corrplot because the namespace may be masked by pls::corrplot
corrplot::corrplot(
  correl3,
  method = "circle",
  type = "upper",
  order = "hclust",
  tl.cex = 0.5,
  main = paste("\nClustered correlations of reduced fingerprint matrix (ncol=",
               ncol(correl3),") where abs(corr) < ", round(cutoff,4))
)
```



Correlation grid #2

The above correlation grid does not display as many clusters indicating variables that are highly correlated with each other, which should remove the multicollinearlty problem.

(c) Split, pre-process, and tune

pre-process the data The values for permeability are all positive.

In order to ensure that we do not obtain any negative predictions,
##### we will fit $\log(permeability)$ and then exponentiate the results of the fitting.

```
set.seed(12345)
trainRow <- createDataPartition(log_permeability, p=0.8, list=FALSE)
ctrl <- trainControl(method = "cv")


fingerprints.train     <- fingerprints_filtered3[trainRow, ]
permeability.train     <- permeability[trainRow, ]
log_permeability.train <- log_permeability[trainRow, ]
fingerprints.test      <- fingerprints_filtered3[-trainRow, ]
permeability.test      <- permeability[-trainRow, ]
log_permeability.test  <- log_permeability[-trainRow, ]
```

Now split the reduced data into a training and a test set,

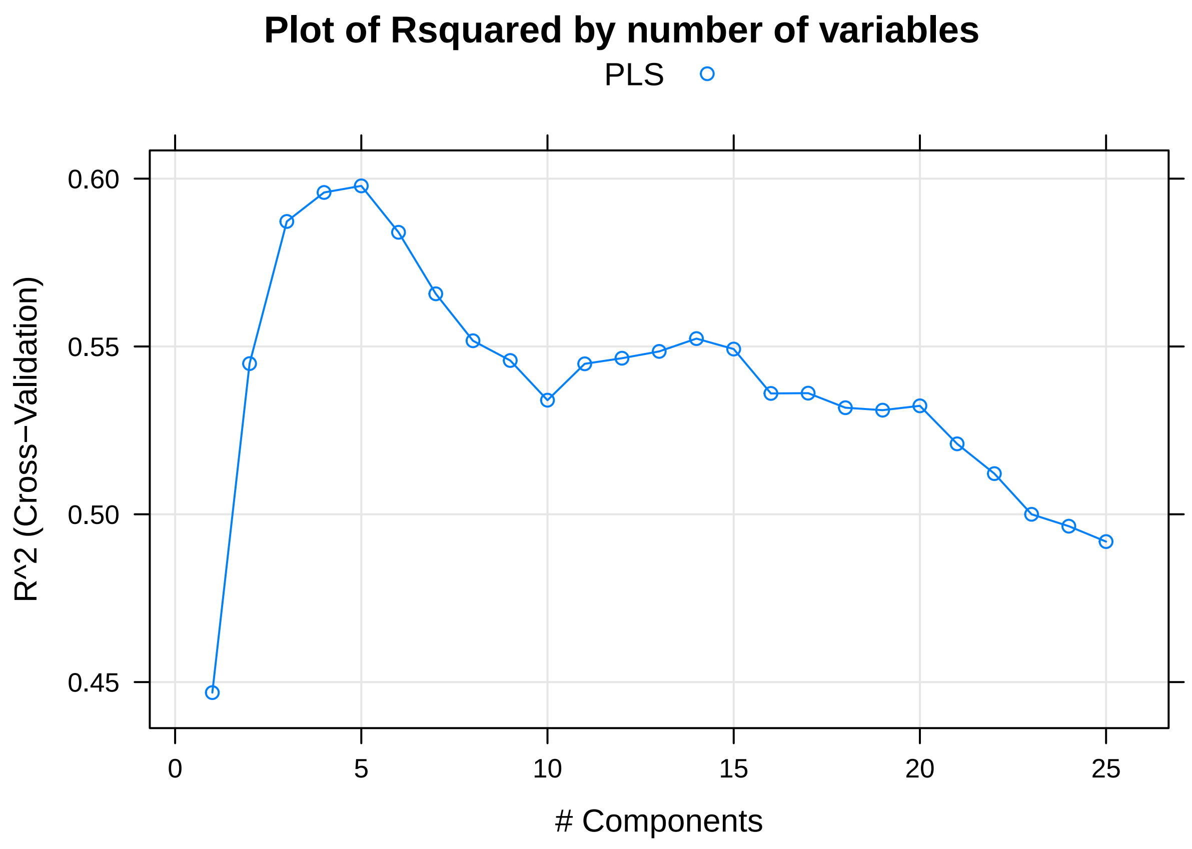
```
#library(pls)
## Run PLS
set.seed(100)
plsTune <- train(x = fingerprints.train,
                 y = log_permeability.train,
                 method = "pls",
                 metric='Rsquared',
                 tuneLength = 25,
                 tuneGrid = expand.grid(ncomp = 1:25),
                 trControl = ctrl,
                 preProcess=c('center', 'scale')
                 )
plsTune
```

and tune a PLS model.

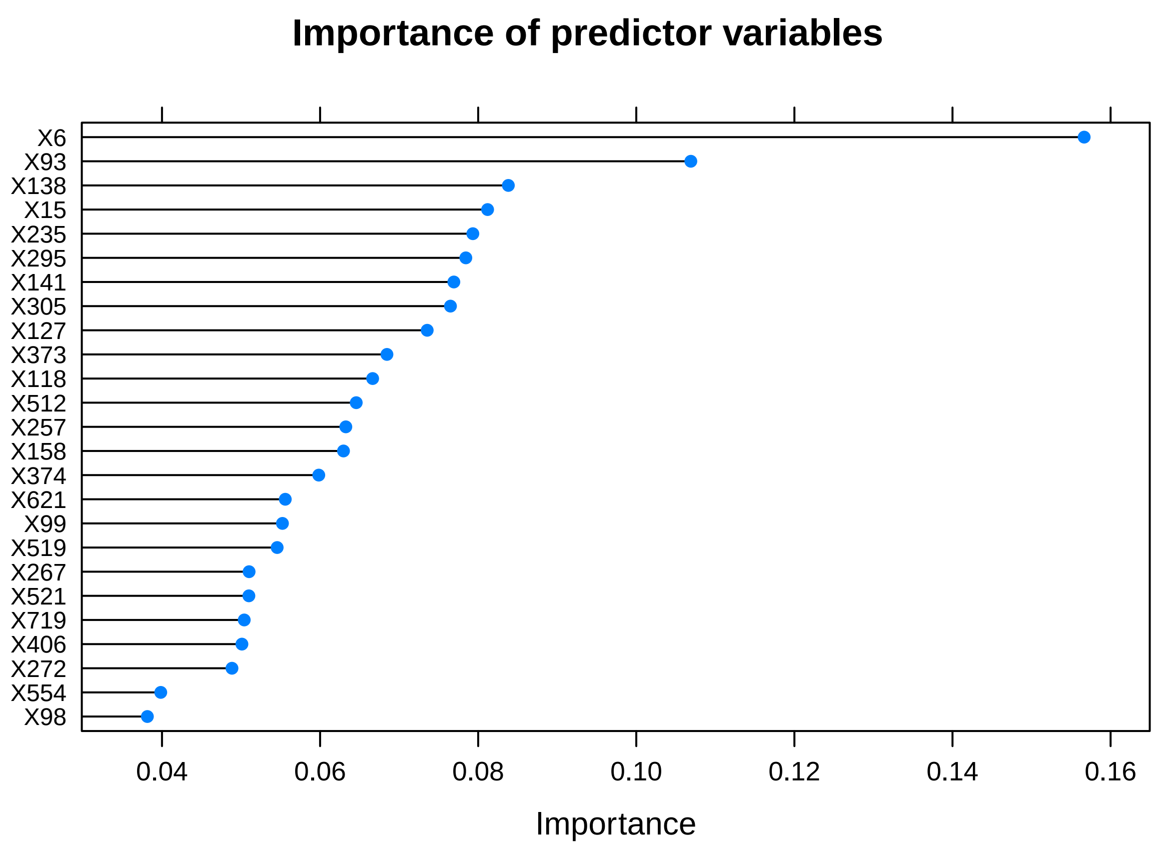
```
## Partial Least Squares
##
## 133 samples
##  70 predictor
##
## Pre-processing: centered (70), scaled (70)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 120, 120, 119, 121, 120, 119, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared   MAE
##    1     1.129601  0.4468554  0.9303761
##    2     1.074201  0.5448874  0.8641737
##    3     1.035358  0.5872487  0.8454854
##    4     1.023068  0.5958865  0.8085643
##    5     1.015558  0.5978475  0.8150107
##    6     1.034142  0.5840170  0.8297577
##    7     1.058789  0.5656890  0.8495580
##    8     1.082135  0.5516989  0.8697853
##    9     1.097500  0.5458186  0.8811166
##   10     1.110163  0.5339912  0.8972611
##   11     1.096836  0.5448363  0.8925973
##   12     1.094091  0.5464926  0.8881003
##   13     1.096872  0.5485320  0.8813345
##   14     1.091911  0.5523404  0.8805000
##   15     1.104452  0.5492240  0.8893918
##   16     1.125506  0.5360055  0.9159197
##   17     1.128329  0.5360832  0.9165264
##   18     1.132735  0.5317496  0.9149671
##   19     1.134723  0.5310171  0.9105706
##   20     1.133074  0.5323104  0.9099273
##   21     1.150676  0.5209886  0.9314301
##   22     1.166670  0.5121152  0.9443104
##   23     1.186409  0.4999861  0.9565978
##   24     1.194800  0.4964499  0.9626836
##   25     1.203732  0.4918587  0.9702770
```

```
##
## Rsquared was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 5.
```

```
plsResamples <- plsTune$results
plsResamples$Model <- "PLS"

xyplot(Rsquared ~ ncomp,
       data = plsResamples,
       #aspect = 1,
       xlab = "# Components",
       ylab = "R^2 (Cross-Validation)",
       auto.key = list(),
       groups = Model,
       type = c("o", "g"),
       main="Plot of Rsquared by number of variables")
```



```
#### Importance plot of predictor variables
plsImp <- varImp(plsTune, scale = FALSE)
plot(plsImp, top = 25,
     scales = list(y = list(cex = .75)),
     main="Importance of predictor variables")
```



```
plsTune$bestTune$ncomp
```

How many latent variables are optimal and what is the corresponding resampled estimate of R2?

```
## [1] 5
```

```
plsTune$results[plsTune$bestTune$ncomp,]
```

```
##   ncomp     RMSE  Rsquared       MAE    RMSESD RsquaredSD    MAESD
## 5     5 1.015558 0.5978475 0.8150107 0.2109767  0.1541557 0.149285
```

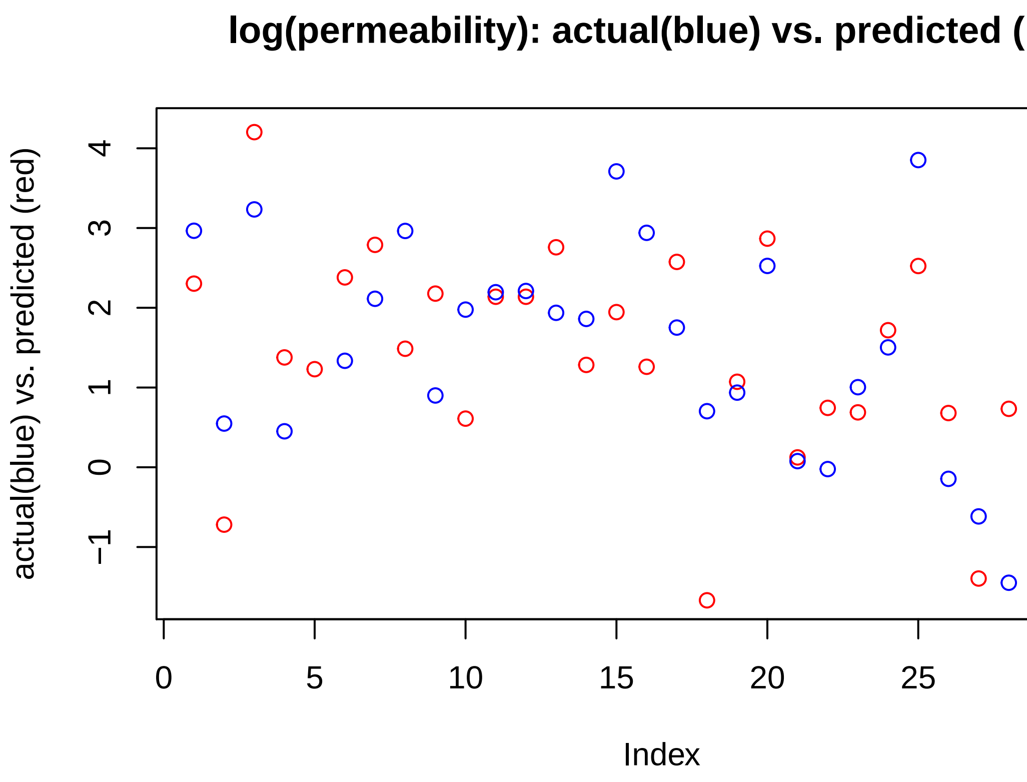
The optimal number of latent variables is 5 and the corresponding resampled estimate of R^2 is 0.5978475 .

(d) Predict the response for the test set.

```r
#### We have predicted the log of permeability
log_pls_test_y_hat <- predict(object = plsTune, newdata = fingerprints.test    )
log_pls_test_stats <- postResample(pred = log_pls_test_y_hat, obs = log_permeability.test)
(log_pls_test_stats <- rbind(log_pls_test_stats)) %>%
  kable() %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

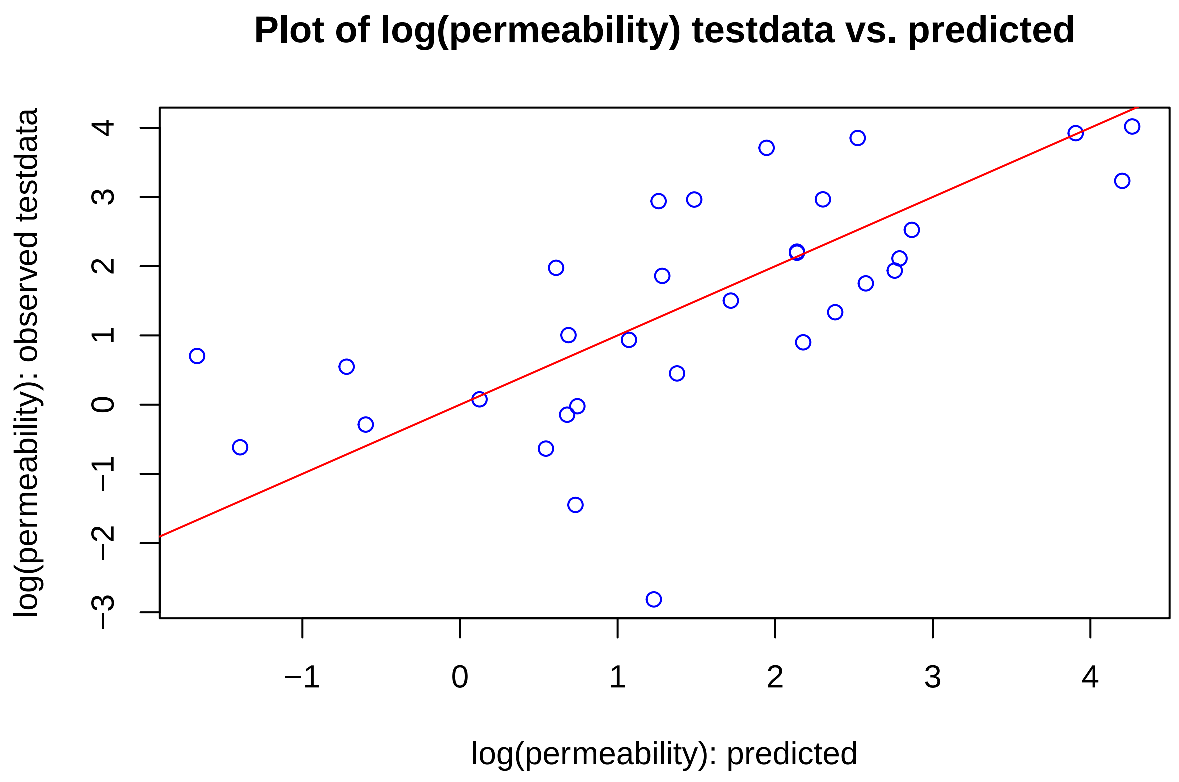
	RMSE	Rsquared	MAE
log_pls_test_stats	1.261313	0.4465345	0.9552875

```r
# Plot actual and predicted permeability by index
plot(log_pls_test_y_hat,col="red",
     ylab="actual(blue) vs. predicted (red)",
     main="log(permeability): actual(blue) vs. predicted (red)")
points(log_permeability.test,col="blue")
```



Predict the log(permeability)

```r
#### Plot of log(observed) vs. log(predicted)
main=paste("Plot of log(permeability)",
            "testdata vs. predicted")
plot(log_permeability.test~log_pls_test_y_hat,
     main=main,col="blue",
     ylab="log(permeability): observed testdata",
     xlab="log(permeability): predicted")
abline(a=0,b=1,col="red")
```

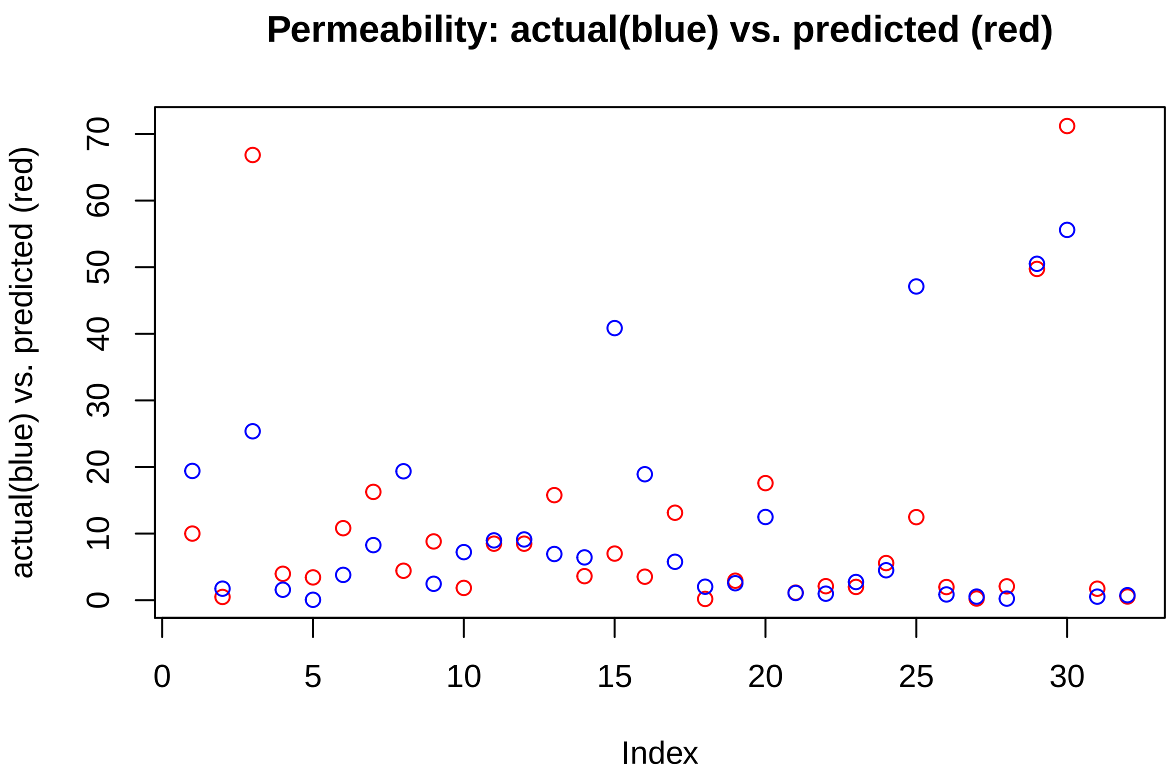


```
pls_test_y_hat <- exp(log_pls_test_y_hat)
pls_test_stats <- postResample(pred = pls_test_y_hat, obs = permeability.test)
(pls_test_stats <- rbind(pls_test_stats)) %>%
  kable() %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

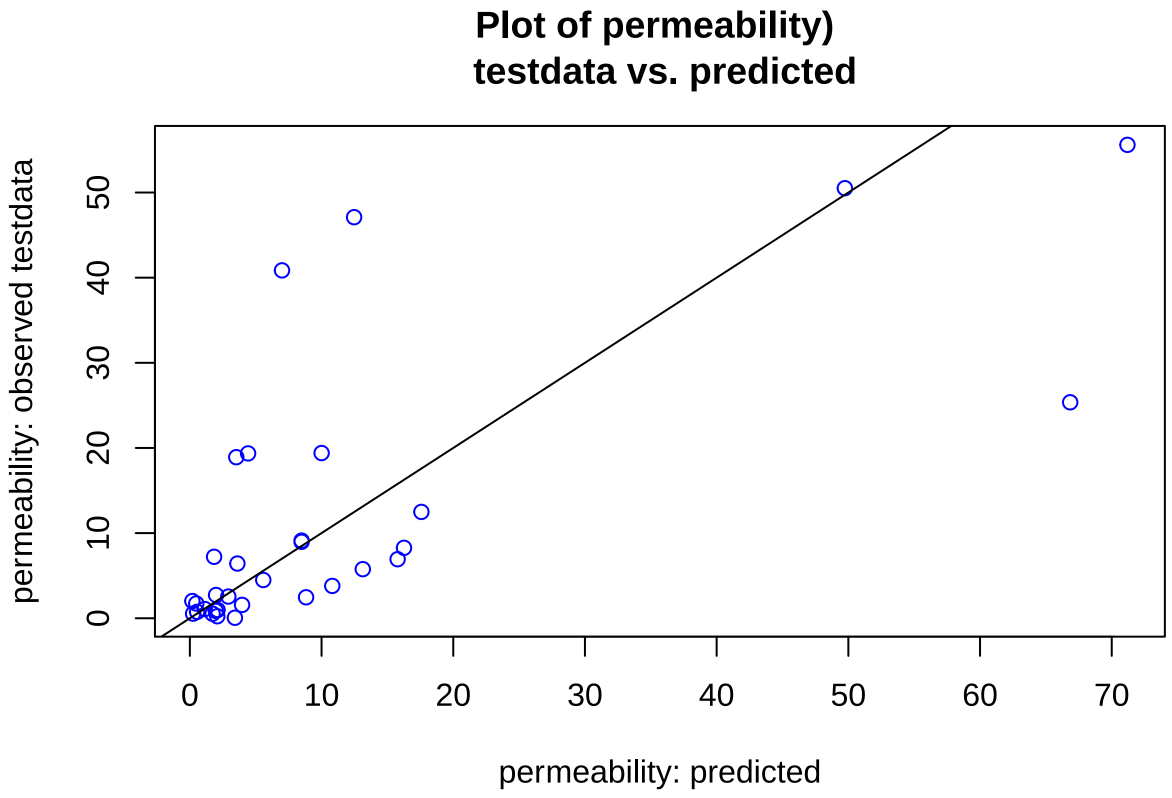
	RMSE	Rsquared	MAE
pls_test_stats	12.79551	0.4969029	7.340892

```
# Plot actual and predicted permeability by index
plot(pls_test_y_hat,col="red",
     ylab="actual(blue) vs. predicted (red)",
     main="Permeability: actual(blue) vs. predicted (red)")
points(permeability.test,col="blue")
```

Because we predicted the log of the permeability, exponentiate to get the genuine value

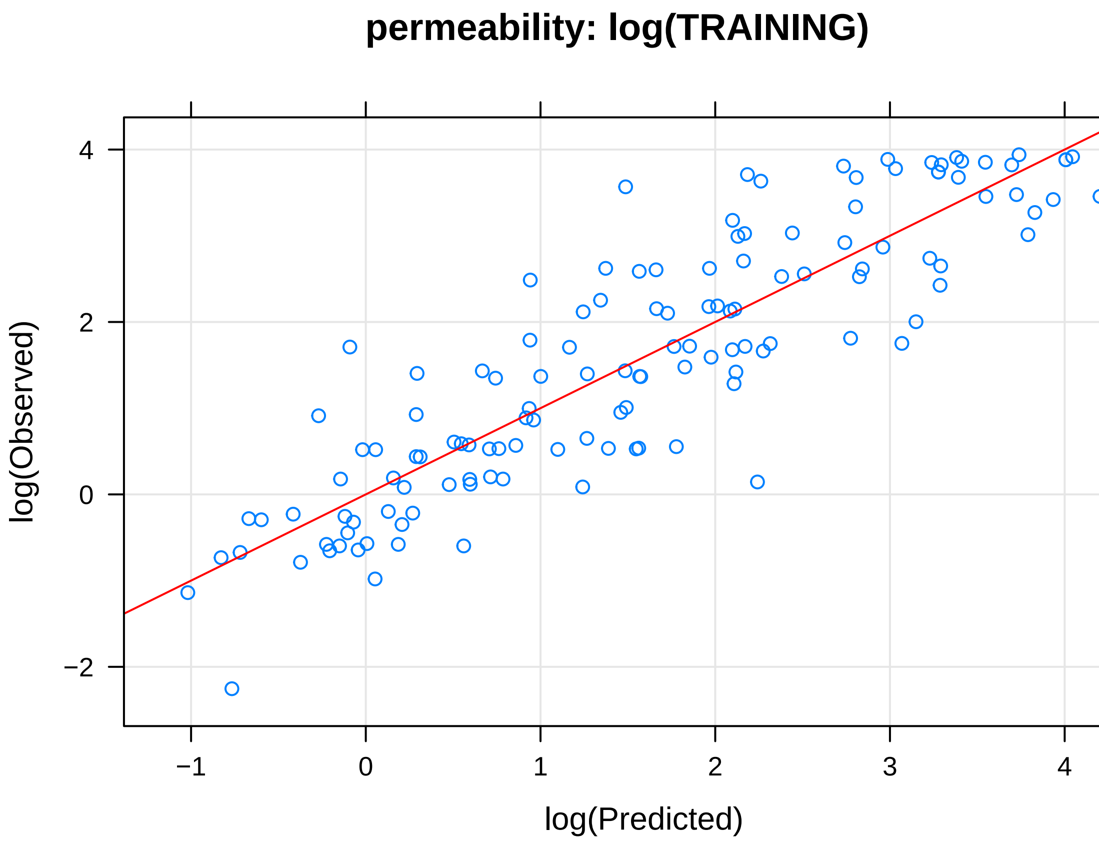


```
main=paste("Plot of permeability)","\n",
            "testdata vs. predicted")
plot(permeability.test~ pls_test_y_hat,
     main=main,col="blue",
     ylab="permeability: observed testdata",
     xlab="permeability: predicted")
abline(a=0,b=1)
```



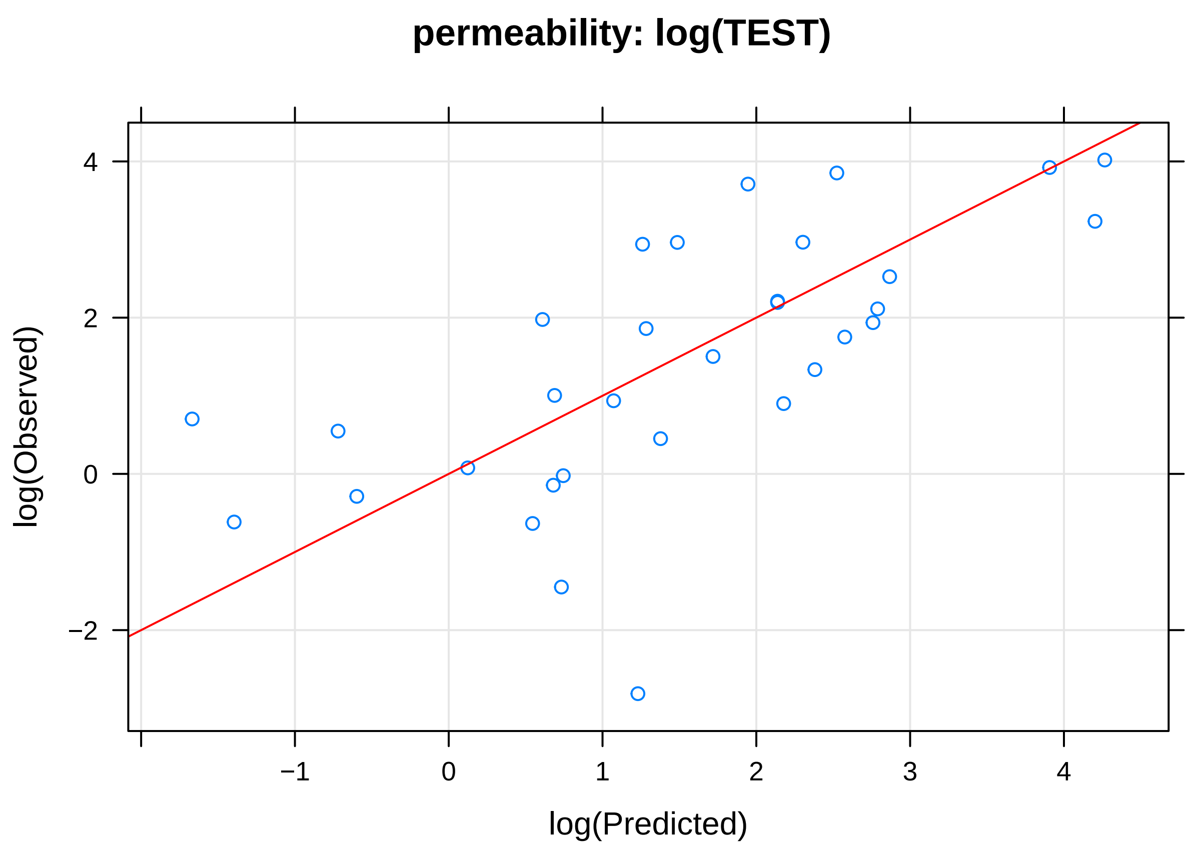
What is the test set estimate of R2? The test set estimate of R^2 is 0.4969029 .

```
par(mfrow=c(1,2))
xyplot(log_permeability.train ~ predict(plsTune),
## plot the points (type = 'p') and a background grid ('g')
type = c("p", "g"),
xlab = "log(Predicted)", ylab = "log(Observed)", main="permeability: log(TRAINING)",
panel = function(x,y, ...){
  panel.xyplot(x,y, ...)
  panel.abline(a=0,b=1,col="red")
  }
)
```

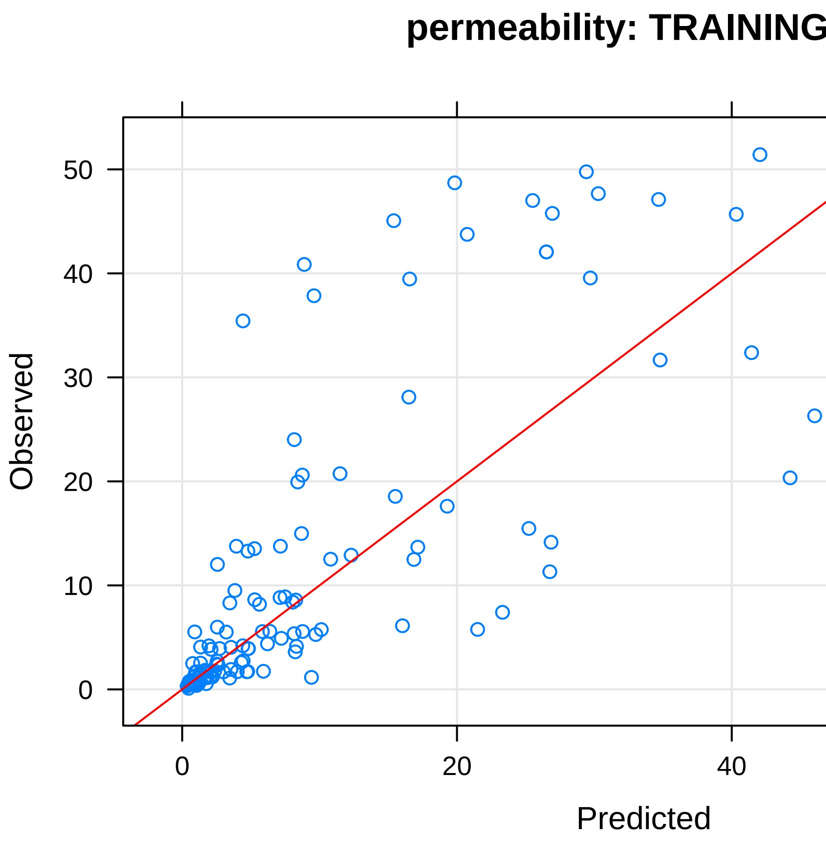


Plot results in log space

```
xyplot(log_permeability.test ~ predict(plsTune,newdata = fingerprints.test),
## plot the points (type = 'p') and a background grid ('g')
type = c("p", "g"),
xlab = "log(Predicted)", ylab = "log(Observed)", main="permeability: log(TEST)",
panel = function(x,y, ...){
  panel.xyplot(x,y, ...)
  panel.abline(a=0,b=1,col="red")
  }
)
```

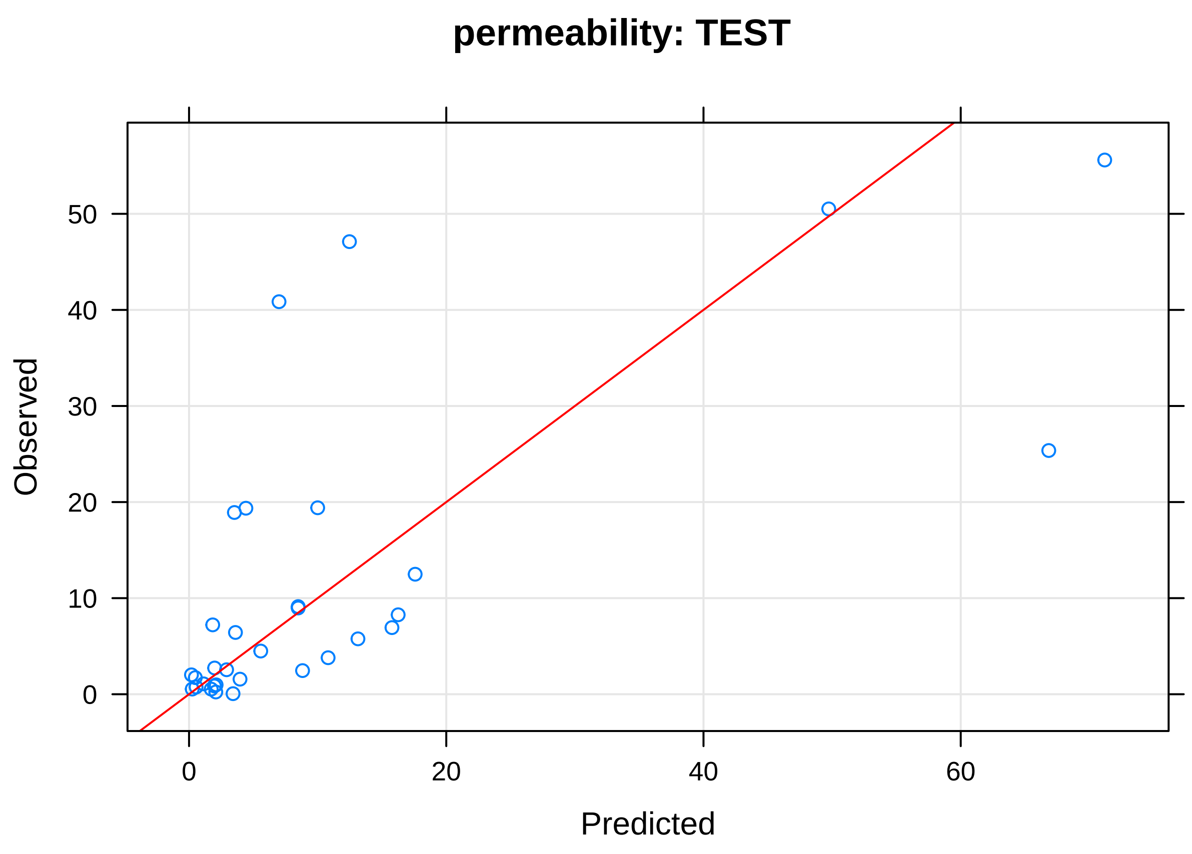

permeability: log(TEST)
log(Observed)
log(Predicted)
4
2
0
−2
−1
0
1
2
3
4

```
par(mfrow=c(1,2))
xyplot(permeability.train ~ exp(predict(plsTune)),
## plot the points (type = 'p') and a background grid ('g')
type = c("p", "g"),
xlab = "Predicted", ylab = "Observed", main="permeability: TRAINING",
panel = function(x,y, ...){
  panel.xyplot(x,y, ...)
  panel.abline(a=0,b=1,col="red")
  }
)
```

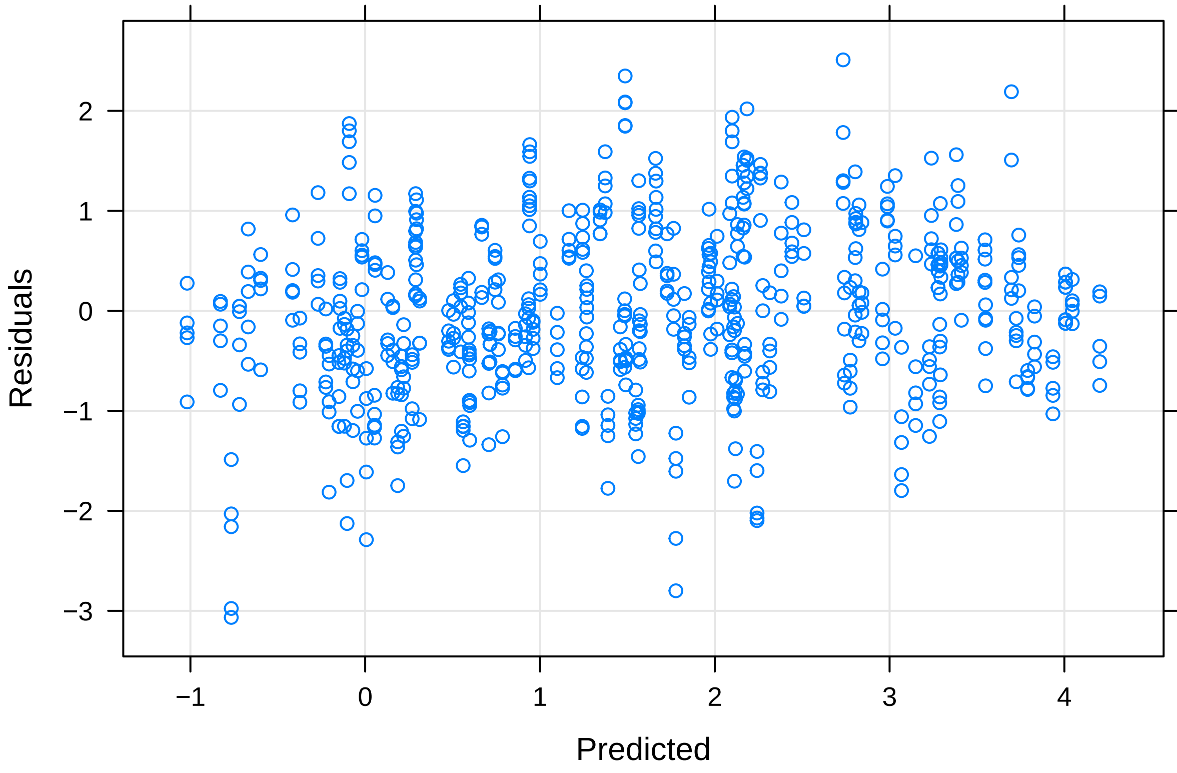


Plot results transformed back from log space

```
xyplot(permeability.test ~ exp(predict(plsTune,newdata = fingerprints.test)),
## plot the points (type = 'p') and a background grid ('g')
type = c("p", "g"),
xlab = "Predicted", ylab = "Observed", main="permeability: TEST",
panel = function(x,y, ...){
  panel.xyplot(x,y, ...)
  panel.abline(a=0,b=1,col="red")
  }
)
```



```r
xyplot(resid(plsTune) ~ predict(plsTune),
type = c("p", "g"),
xlab = "Predicted", ylab = "Residuals")
```



Plot the residuals

(e) Try building other models discussed in this chapter.

```
set.seed(100)
pcrTune <- train(x = fingerprints.train,
                 y = log_permeability.train,
                 method = "pcr",
                 metric='Rsquared',
                 tuneLength = 30,
                 tuneGrid = expand.grid(ncomp = 1:30),
                 trControl = ctrl,
                 preProcess=c('center', 'scale')
                 )
pcrTune
```

Principal Components Regression

```
## Principal Component Analysis
##
## 133 samples
##  70 predictor
##
## Pre-processing: centered (70), scaled (70)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 120, 120, 119, 121, 120, 119, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared   MAE
##    1     1.396449  0.1921303  1.1885866
##    2     1.407526  0.1628107  1.2076770
##    3     1.258203  0.3136725  1.0753989
##    4     1.249699  0.3191919  1.0674092
##    5     1.216096  0.3547920  1.0347397
##    6     1.202460  0.3684778  1.0040539
##    7     1.196000  0.3741053  1.0057957
##    8     1.199824  0.3725864  1.0065136
##    9     1.193223  0.3798376  0.9928497
##   10     1.156392  0.4273598  0.9409841
##   11     1.148142  0.4458268  0.9293701
##   12     1.149307  0.4418021  0.9395009
##   13     1.189950  0.4174143  0.9775562
##   14     1.169940  0.4330941  0.9583958
##   15     1.150940  0.4485630  0.9324270
##   16     1.136926  0.4717093  0.9122760
##   17     1.146646  0.4626980  0.9219600
##   18     1.156057  0.4440877  0.9343078
##   19     1.142685  0.4600692  0.9313502
##   20     1.135221  0.4660711  0.9356840
##   21     1.109920  0.4964814  0.9055208
##   22     1.092821  0.5137321  0.8943207
##   23     1.090298  0.5178388  0.8911599
##   24     1.079791  0.5346728  0.8942976
##   25     1.083962  0.5354908  0.8969697
##   26     1.065363  0.5490485  0.8753033
##   27     1.025594  0.5874146  0.8185133
##   28     1.000658  0.6116411  0.7915114
##   29     0.997372  0.6158315  0.7872849
##   30     1.038243  0.5864712  0.8231064
##
## Rsquared was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 29.
```

```
pcrTune$bestTune$ncomp
```

```
## [1] 29
```

```
pcrTune$results[pcrTune$bestTune$ncomp,]
```

```
##    ncomp     RMSE  Rsquared       MAE    RMSESD RsquaredSD     MAESD
## 29    29 0.997372 0.6158315 0.7872849 0.1835222  0.1404608 0.1290529
```

```
log_pcr_test_y_hat <- predict(object = pcrTune, newdata = fingerprints.test   )
log_pcr_test_stats <- postResample(pred = log_pcr_test_y_hat, obs = log_permeability.test)
(log_pcr_test_stats <- rbind(log_pcr_test_stats))
```

```
##                        RMSE  Rsquared       MAE
## log_pcr_test_stats 1.222122 0.4707882 0.9254839
```

```
pcr_test_y_hat <- exp(log_pcr_test_y_hat)
pcr_test_stats <- postResample(pred = pcr_test_y_hat, obs = permeability.test)
(pcr_test_stats <- rbind(pcr_test_stats)) %>%
  kable() %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

	RMSE	Rsquared	MAE
pcr_test_stats	11.05509	0.5091551	6.912708

```r
ridgeGrid <- data.frame(.lambda = seq(0, 1, length = 101))
set.seed(100)
ridgeTune <- train(x = fingerprints.train,
                   y = log_permeability.train,
                   method = "ridge",
                   metric='Rsquared',
## Fit the model over many penalty values
                   tuneGrid = ridgeGrid,
                   trControl = ctrl,
## put the predictors on the same scale
                   preProc = c("center", "scale"))
ridgeTune
```

Ridge Regression

```
## Ridge Regression
##
## 133 samples
##  70 predictor
##
## Pre-processing: centered (70), scaled (70)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 120, 120, 119, 121, 120, 119, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE      Rsquared   MAE
##   0.00    2.397076  0.4316102  1.3197444
##   0.01    1.071955  0.5584276  0.8561084
##   0.02    1.052839  0.5718695  0.8396762
##   0.03    1.045033  0.5778364  0.8378414
##   0.04    1.040172  0.5819314  0.8362268
##   0.05    1.036598  0.5852476  0.8341247
##   0.06    1.033790  0.5881104  0.8320177
##   0.07    1.031524  0.5906508  0.8297302
##   0.08    1.029681  0.5929363  0.8273909
##   0.09    1.028187  0.5950079  0.8254853
##   0.10    1.026992  0.5968944  0.8245605
##   0.11    1.026057  0.5986173  0.8235870
##   0.12    1.025354  0.6001937  0.8227112
##   0.13    1.024859  0.6016377  0.8218256
##   0.14    1.024552  0.6029610  0.8213229
##   0.15    1.024417  0.6041740  0.8210594
##   0.16    1.024440  0.6052854  0.8207994
##   0.17    1.024610  0.6063031  0.8205479
##   0.18    1.024915  0.6072341  0.8203080
##   0.19    1.025346  0.6080846  0.8200821
##   0.20    1.025896  0.6088602  0.8198715
##   0.21    1.026558  0.6095661  0.8196772
##   0.22    1.027324  0.6102069  0.8194994
##   0.23    1.028190  0.6107869  0.8195686
##   0.24    1.029149  0.6113101  0.8201114
##   0.25    1.030198  0.6117800  0.8206684
##   0.26    1.031331  0.6122001  0.8212393
##   0.27    1.032546  0.6125733  0.8218234
##   0.28    1.033838  0.6129027  0.8225941
##   0.29    1.035203  0.6131909  0.8233837
##   0.30    1.036639  0.6134403  0.8242762
##   0.31    1.038143  0.6136533  0.8254678
##   0.32    1.039712  0.6138321  0.8266864
##   0.33    1.041344  0.6139787  0.8279656
##   0.34    1.043035  0.6140950  0.8292575
##   0.35    1.044785  0.6141828  0.8307187
##   0.36    1.046591  0.6142437  0.8321818
##   0.37    1.048450  0.6142793  0.8336465
##   0.38    1.050361  0.6142911  0.8351124
##   0.39    1.052323  0.6142804  0.8365792
##   0.40    1.054333  0.6142486  0.8381878
##   0.41    1.056390  0.6141968  0.8399258
##   0.42    1.058492  0.6141263  0.8417548
##   0.43    1.060639  0.6140381  0.8437206
##   0.44    1.062827  0.6139331  0.8458216
##   0.45    1.065057  0.6138125  0.8479638
##   0.46    1.067327  0.6136770  0.8501555
##   0.47    1.069636  0.6135275  0.8524413
##   0.48    1.071982  0.6133649  0.8547803
##   0.49    1.074364  0.6131898  0.8571119
##   0.50    1.076782  0.6130030  0.8594360
##   0.51    1.079234  0.6128053  0.8619891
##   0.52    1.081719  0.6125971  0.8646629
##   0.53    1.084237  0.6123791  0.8673909
##   0.54    1.086785  0.6121519  0.8701112
##   0.55    1.089364  0.6119161  0.8728313
##   0.56    1.091973  0.6116721  0.8755409
##   0.57    1.094610  0.6114204  0.8782401
##   0.58    1.097276  0.6111615  0.8809289
##   0.59    1.099968  0.6108958  0.8836074
##   0.60    1.102686  0.6106237  0.8863648
##   0.61    1.105430  0.6103457  0.8891539
##   0.62    1.108199  0.6100620  0.8919939
##   0.63    1.110992  0.6097730  0.8948208
##   0.64    1.113808  0.6094791  0.8976349
##   0.65    1.116647  0.6091806  0.9004362
##   0.66    1.119508  0.6088777  0.9032249
##   0.67    1.122390  0.6085708  0.9060013
##   0.68    1.125293  0.6082601  0.9087654
##   0.69    1.128217  0.6079458  0.9115174
##   0.70    1.131160  0.6076283  0.9142657
##   0.71    1.134122  0.6073077  0.9171158
##   0.72    1.137102  0.6069843  0.9199536
##   0.73    1.140101  0.6066583  0.9227791
##   0.74    1.143117  0.6063298  0.9255924
##   0.75    1.146150  0.6059990  0.9283938
##   0.76    1.149199  0.6056662  0.9311832
##   0.77    1.152264  0.6053315  0.9339609
##   0.78    1.155345  0.6049950  0.9367269
##   0.79    1.158441  0.6046569  0.9394814
##    0.80    1.161551  0.6043174  0.9422244
##    0.81    1.164675  0.6039765  0.9449562
##    0.82    1.167813  0.6036345  0.9476767
##    0.83    1.170965  0.6032914  0.9503861
##    0.84    1.174129  0.6029474  0.9530844
##    0.85    1.177305  0.6026025  0.9558216
##    0.86    1.180494  0.6022569  0.9585769
##    0.87    1.183695  0.6019107  0.9613213
##    0.88    1.186906  0.6015639  0.9640549
##    0.89    1.190129  0.6012167  0.9667778
##    0.90    1.193362  0.6008692  0.9694900
##    0.91    1.196605  0.6005214  0.9721917
##    0.92    1.199859  0.6001734  0.9748830
##    0.93    1.203122  0.5998253  0.9776347
##    0.94    1.206394  0.5994772  0.9803794
##    0.95    1.209675  0.5991290  0.9831136
##    0.96    1.212965  0.5987810  0.9858681
##    0.97    1.216263  0.5984331  0.9887185
##    0.98    1.219569  0.5980855  0.9915969
##    0.99    1.222882  0.5977381  0.9944956
##    1.00    1.226204  0.5973910  0.9973876
##
## Rsquared was used to select the optimal model using the largest value.
## The final value used for the model was lambda = 0.38.
```

```
ridgeTune$bestTune$lambda
```

```
## [1] 0.38
```

```
ridgeTune$results[rownames(ridgeTune$bestTune),]
```

```
##    lambda     RMSE  Rsquared       MAE    RMSESD RsquaredSD     MAESD
## 39   0.38 1.050361 0.6142911 0.8351124 0.2394788  0.1645513 0.1630461
```

```
log_ridge_test_y_hat <- predict(object = ridgeTune, newdata = fingerprints.test    )
log_ridge_test_stats <- postResample(pred = log_ridge_test_y_hat, obs = log_permeability.test)
(log_ridge_test_stats <- rbind(log_ridge_test_stats))
```

```
##                          RMSE  Rsquared       MAE
## log_ridge_test_stats 1.245917 0.5000962 0.9837901
```

```
ridge_test_y_hat <- exp(log_ridge_test_y_hat)
ridge_test_stats <- postResample(pred = ridge_test_y_hat, obs = permeability.test)
(ridge_test_stats <- rbind(ridge_test_stats)) %>%
  kable() %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

	RMSE	Rsquared	MAE
ridge_test_stats	17.63038	0.5331787	10.12143

```
enetGrid = expand.grid(.lambda  =seq(0,    1,   length=21),
                       .fraction=seq(0.05, 1.0, length=20))
set.seed(100)
enetTune <- train(x = fingerprints.train,
                  y = log_permeability.train,
                  method = "enet",
                  metric='Rsquared',
                  tuneGrid = enetGrid,
                  trControl = ctrl,
                  preProc = c("center", "scale")
                  )
# enetTune
## printing suppressed because of length of results:

## Rsquared was used to select the optimal model using the largest value.
## The final values used for the model were fraction = 0.5 and lambda = 0.45.

enetTune$bestTune
```

Elasticnet

```
##     fraction lambda
## 190      0.5   0.45
```

```
enetTune$results[rownames(enetTune$bestTune),]
```

```
##     lambda fraction      RMSE  Rsquared       MAE    RMSESD RsquaredSD
## 190   0.45      0.5 0.9782374 0.6339662 0.7726211 0.1649071  0.1244465
##         MAESD
## 190 0.1105195
```

```
log_enet_test_y_hat <- predict(object = enetTune, newdata = fingerprints.test   )
log_enet_test_stats <- postResample(pred = log_enet_test_y_hat, obs = log_permeability.test)
(log_enet_test_stats <- rbind(log_enet_test_stats))
```

```
##                         RMSE  Rsquared       MAE
## log_enet_test_stats 1.153593 0.5134051 0.8594702
```

```
enet_test_y_hat <- exp(log_enet_test_y_hat)
enet_test_stats <- postResample(pred = enet_test_y_hat, obs = permeability.test)
(enet_test_stats <- rbind(enet_test_stats)) %>%
  kable() %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

	RMSE	Rsquared	MAE
enet_test_stats	12.90858	0.4408179	7.388419

```r
plot(enetTune, main="ElasticNet", sub="Maximum R^2 occurs at lambda=0.45 and fraction=0.50")
```

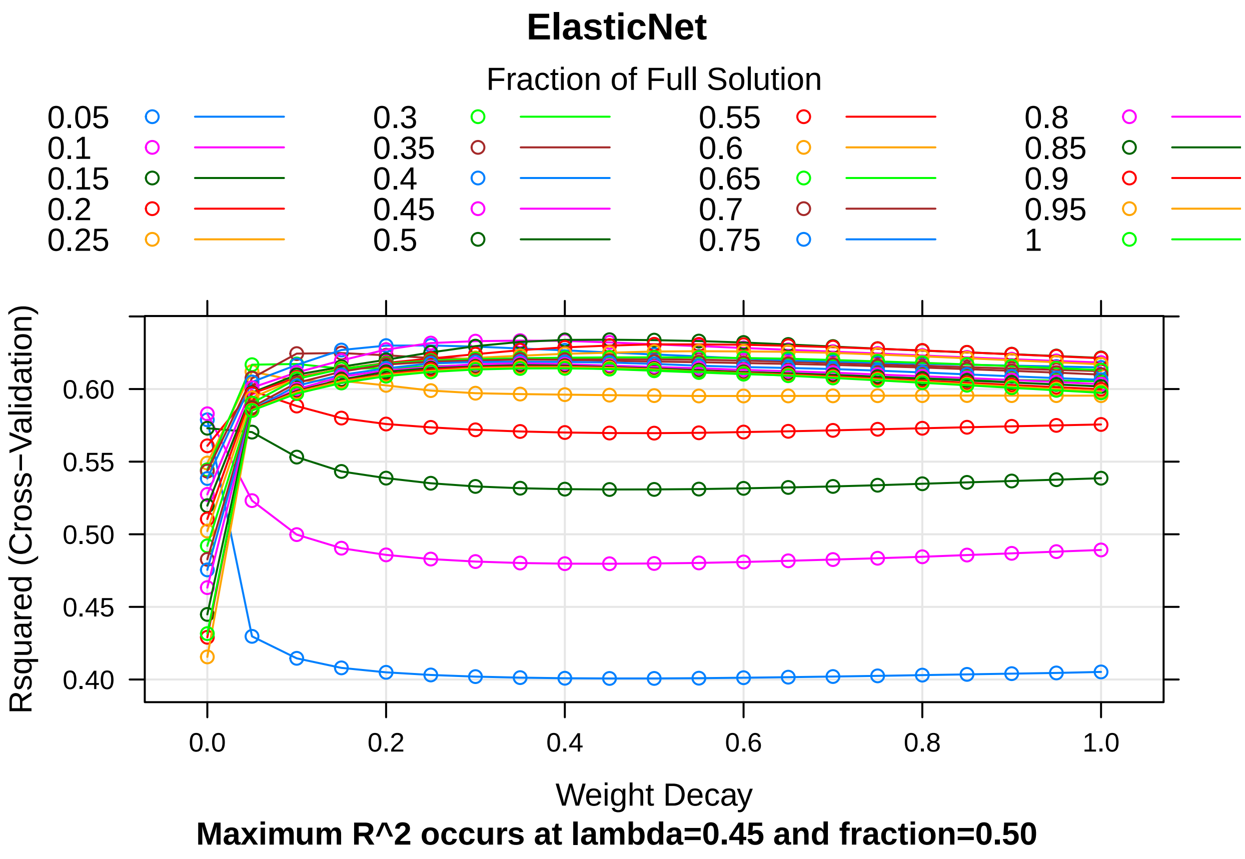


Table 1: Summary of results

	RMSE	Rsquared	MAE
pls_test_stats	12.79551	0.4969029	7.340892
pcr_test_stats	11.05509	0.5091551	6.912708
ridge_test_stats	17.63038	0.5331787	10.121428
enet_test_stats	12.90858	0.4408179	7.388419

```
rbind(pls_test_stats,pcr_test_stats,ridge_test_stats,enet_test_stats ) %>%
  kable(caption = "Summary of results") %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

Do any have better predictive performance? **Ridge** has a better R^2, but this result corresponds to a worse RMSE and MAE.
Using the criterion of maximizing R^2, the associated RMSE and MAE are better on **PCR** than PLS.

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

No, I don't believe that the predictive power from these models are strong enough to replace the laboratory experiment.

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 data set has been renamed
data(ChemicalManufacturingProcess)
# save a copy
origChemicalManufacturingProcess <- ChemicalManufacturingProcess
# Examine the data
# summary with standard deviation and skewness:
#library(moments)
### Because all the data is numeric, we can change from data.frame to matrix
m_ChemicalManufacturingProcess <- as.matrix(ChemicalManufacturingProcess)
rbind(summary(m_ChemicalManufacturingProcess),
      paste0("StDev  :",round(apply(X = m_ChemicalManufacturingProcess, MARGIN = 2, FUN = sd,na.rm=T),2
      paste0("Skew   :",round(skewness(m_ChemicalManufacturingProcess,na.rm=T),2),"  ")) %>%
      as.table()
```

```
##      Yield       BiologicalMaterial01 BiologicalMaterial02 BiologicalMaterial03
##  Min.   :35.25   Min.   :4.580        Min.   :46.87        Min.   :56.97
##  1st Qu.:38.75   1st Qu.:5.978        1st Qu.:52.68        1st Qu.:64.98
##  Median :39.97   Median :6.305        Median :55.09        Median :67.22
##  Mean   :40.18   Mean   :6.411        Mean   :55.69        Mean   :67.70
##  3rd Qu.:41.48   3rd Qu.:6.870        3rd Qu.:58.74        3rd Qu.:70.43
##  Max.   :46.34   Max.   :8.810        Max.   :64.75        Max.   :78.25
##
##  StDev  :1.85    StDev  :0.71         StDev  :4.03         StDev  :4
##  Skew   :0.31    Skew   :0.28         Skew   :0.25         Skew   :0.03
##  BiologicalMaterial04 BiologicalMaterial05 BiologicalMaterial06
##  Min.   : 9.38        Min.   :13.24        Min.   :40.60
##  1st Qu.:11.24        1st Qu.:17.23        1st Qu.:46.05
##  Median :12.10        Median :18.49        Median :48.46
##  Mean   :12.35        Mean   :18.60        Mean   :48.91
##  3rd Qu.:13.22        3rd Qu.:19.90        3rd Qu.:51.34
##  Max.   :23.09        Max.   :24.85        Max.   :59.38
##
##  StDev  :1.77         StDev  :1.84         StDev  :3.75
##  Skew   :1.75         Skew   :0.31         Skew   :0.37
##  BiologicalMaterial07 BiologicalMaterial08 BiologicalMaterial09
##  Min.   :100.0        Min.   :15.88        Min.   :11.44
##  1st Qu.:100.0        1st Qu.:17.06        1st Qu.:12.60
##  Median :100.0        Median :17.51        Median :12.84
##  Mean   :100.0        Mean   :17.49        Mean   :12.85
##  3rd Qu.:100.0        3rd Qu.:17.88        3rd Qu.:13.13
##  Max.   :100.8        Max.   :19.14        Max.   :14.08
##
##  StDev  :0.11         StDev  :0.68         StDev  :0.42
##  Skew   :7.46         Skew   :0.22         Skew   :-0.27
##  BiologicalMaterial10 BiologicalMaterial11 BiologicalMaterial12
##  Min.   :1.770        Min.   :135.8        Min.   :18.35
##  1st Qu.:2.460        1st Qu.:143.8        1st Qu.:19.73
##  Median :2.710        Median :146.1        Median :20.12
##  Mean   :2.801        Mean   :147.0        Mean   :20.20
##  3rd Qu.:2.990        3rd Qu.:149.6        3rd Qu.:20.75
##  Max.   :6.870        Max.   :158.7        Max.   :22.21
##
##  StDev  :0.6          StDev  :4.82         StDev  :0.77
##  Skew   :2.42         Skew   :0.36         Skew   :0.31
##  ManufacturingProcess01 ManufacturingProcess02 ManufacturingProcess03
##  Min.   : 0.00          Min.   : 0.00          Min.   :1.47
##  1st Qu.:10.80          1st Qu.:19.30          1st Qu.:1.53
##  Median :11.40          Median :21.00          Median :1.54
##  Mean   :11.21          Mean   :16.68          Mean   :1.54
##  3rd Qu.:12.15          3rd Qu.:21.50          3rd Qu.:1.55
##  Max.   :14.10          Max.   :22.50          Max.   :1.60
##  NA's   :1              NA's   :3              NA's   :15
##  StDev  :1.82           StDev  :8.47           StDev  :0.02
##  Skew   :-3.95          Skew   :-1.44          Skew   :-0.48
##  ManufacturingProcess04 ManufacturingProcess05 ManufacturingProcess06
##  Min.   :911.0          Min.   : 923.0         Min.   :203.0
##  1st Qu.:928.0          1st Qu.: 986.8         1st Qu.:205.7
##  Median :934.0          Median : 999.2         Median :206.8
##  Mean   :931.9          Mean   :1001.7         Mean   :207.4
##  3rd Qu.:936.0          3rd Qu.:1008.9         3rd Qu.:208.7
##  Max.   :946.0          Max.   :1175.3         Max.   :227.4
##  NA's   :1              NA's   :1              NA's   :2
##  StDev  :6.27           StDev  :30.53          StDev  :2.7
##  Skew   :-0.7           Skew   :2.61           Skew   :3.07
##  ManufacturingProcess07 ManufacturingProcess08 ManufacturingProcess09
##  Min.   :177.0          Min.   :177.0          Min.   :38.89
##  1st Qu.:177.0          1st Qu.:177.0          1st Qu.:44.89
##  Median :177.0          Median :178.0          Median :45.73
##  Mean   :177.5          Mean   :177.6          Mean   :45.66
##  3rd Qu.:178.0          3rd Qu.:178.0          3rd Qu.:46.52
##  Max.   :178.0          Max.   :178.0          Max.   :49.36
##  NA's   :1             NA's   :1
##  StDev  :0.5           StDev  :0.5            StDev  :1.55
##  Skew   :0.08          Skew   :-0.22          Skew   :-0.95
##  ManufacturingProcess10 ManufacturingProcess11 ManufacturingProcess12
##  Min.   : 7.500        Min.   : 7.500         Min.   :   0.0
##  1st Qu.: 8.700        1st Qu.: 9.000         1st Qu.:   0.0
##  Median : 9.100        Median : 9.400         Median :   0.0
##  Mean   : 9.179        Mean   : 9.386         Mean   : 857.8
##  3rd Qu.: 9.550        3rd Qu.: 9.900         3rd Qu.:   0.0
##  Max.   :11.600        Max.   :11.500         Max.   :4549.0
##  NA's   :9             NA's   :10             NA's   :1
##  StDev  :0.77          StDev  :0.72           StDev  :1784.53
##  Skew   :0.66          Skew   :-0.02          Skew   :1.59
##  ManufacturingProcess13 ManufacturingProcess14 ManufacturingProcess15
##  Min.   :32.10         Min.   :4701           Min.   :5904
##  1st Qu.:33.90         1st Qu.:4828           1st Qu.:6010
##  Median :34.60         Median :4856           Median :6032
##  Mean   :34.51         Mean   :4854           Mean   :6039
##  3rd Qu.:35.20         3rd Qu.:4882           3rd Qu.:6061
##  Max.   :38.60         Max.   :5055           Max.   :6233
##                        NA's   :1
##  StDev  :1.02          StDev  :54.52          StDev  :58.31
##  Skew   :0.48          Skew   :-0.01          Skew   :0.68
##  ManufacturingProcess16 ManufacturingProcess17 ManufacturingProcess18
##  Min.   :   0          Min.   :31.30          Min.   :   0
##  1st Qu.:4561          1st Qu.:33.50          1st Qu.:4813
##  Median :4588          Median :34.40          Median :4835
##  Mean   :4566          Mean   :34.34          Mean   :4810
##  3rd Qu.:4619          3rd Qu.:35.10          3rd Qu.:4862
##  Max.   :4852          Max.   :40.00          Max.   :4971
##
##  StDev  :351.7         StDev  :1.25           StDev  :367.48
##  Skew   :-12.53        Skew   :1.17           Skew   :-12.85
##  ManufacturingProcess19 ManufacturingProcess20 ManufacturingProcess21
##  Min.   :5890          Min.   :   0           Min.   :-1.8000
##  1st Qu.:6001          1st Qu.:4553           1st Qu.:-0.6000
##  Median :6022          Median :4582           Median :-0.3000
##  Mean   :6028          Mean   :4556           Mean   :-0.1642
##  3rd Qu.:6050          3rd Qu.:4610           3rd Qu.: 0.0000
##  Max.   :6146          Max.   :4759           Max.   : 3.6000
##
##  StDev  :45.58         StDev  :349.01         StDev  :0.78
##  Skew   :0.3           Skew   :-12.75         Skew   :1.74
##  ManufacturingProcess22 ManufacturingProcess23 ManufacturingProcess24
##  Min.   : 0.000        Min.   :0.000          Min.   : 0.000
##  1st Qu.: 3.000        1st Qu.:2.000          1st Qu.: 4.000
##  Median : 5.000        Median :3.000          Median : 8.000
##  Mean   : 5.406        Mean   :3.017          Mean   : 8.834
##  3rd Qu.: 8.000        3rd Qu.:4.000          3rd Qu.:14.000
##  Max.   :12.000        Max.   :6.000          Max.   :23.000
##  NA's   :1             NA's   :1              NA's   :1
##  StDev  :3.33          StDev  :1.66           StDev  :5.8
##  Skew   :0.32          Skew   :0.2            Skew   :0.36
##  ManufacturingProcess25 ManufacturingProcess26 ManufacturingProcess27
##  Min.   :   0           Min.   :   0           Min.   :   0
##  1st Qu.:4832           1st Qu.:6020           1st Qu.:4560
##  Median :4855           Median :6047           Median :4587
##  Mean   :4828           Mean   :6016           Mean   :4563
##  3rd Qu.:4877           3rd Qu.:6070           3rd Qu.:4609
##  Max.   :4990           Max.   :6161           Max.   :4710
##  NA's   :5              NA's   :5              NA's   :5
##  StDev  :373.48         StDev  :464.87         StDev  :353.98
##  Skew   :-12.74         Skew   :-12.78         Skew   :-12.63
##  ManufacturingProcess28 ManufacturingProcess29 ManufacturingProcess30
##  Min.   : 0.000         Min.   : 0.00          Min.   : 0.000
##  1st Qu.: 0.000         1st Qu.:19.70          1st Qu.: 8.800
##  Median :10.400         Median :19.90          Median : 9.100
##  Mean   : 6.592         Mean   :20.01          Mean   : 9.161
##  3rd Qu.:10.750         3rd Qu.:20.40          3rd Qu.: 9.700
##  Max.   :11.500         Max.   :22.00          Max.   :11.200
##  NA's   :5              NA's   :5              NA's   :5
##  StDev  :5.25           StDev  :1.66           StDev  :0.98
##  Skew   :-0.46          Skew   :-10.17         Skew   :-4.8
##  ManufacturingProcess31 ManufacturingProcess32 ManufacturingProcess33
##  Min.   : 0.00          Min.   :143.0          Min.   :56.00
##  1st Qu.:70.10          1st Qu.:155.0          1st Qu.:62.00
##  Median :70.80          Median :158.0          Median :64.00
##  Mean   :70.18          Mean   :158.5          Mean   :63.54
##  3rd Qu.:71.40          3rd Qu.:162.0          3rd Qu.:65.00
##  Max.   :72.50          Max.   :173.0          Max.   :70.00
##  NA's   :5                                     NA's   :5
##  StDev  :5.56           StDev  :5.4            StDev  :2.48
##  Skew   :-11.93         Skew   :0.21           Skew   :-0.13
##  ManufacturingProcess34 ManufacturingProcess35 ManufacturingProcess36
##  Min.   :2.300          Min.   :463.0          Min.   :0.01700
##  1st Qu.:2.500          1st Qu.:490.0          1st Qu.:0.01900
##  Median :2.500          Median :495.0          Median :0.02000
##  Mean   :2.494          Mean   :495.6          Mean   :0.01957
##  3rd Qu.:2.500          3rd Qu.:501.5          3rd Qu.:0.02000
##  Max.   :2.600          Max.   :522.0          Max.   :0.02200
##  NA's   :5              NA's   :5              NA's   :5
##  StDev  :0.05           StDev  :10.82          StDev  :0
##  Skew   :-0.27          Skew   :-0.16          Skew   :0.15
##  ManufacturingProcess37 ManufacturingProcess38 ManufacturingProcess39
##  Min.   :0.000          Min.   :0.000          Min.   :0.000
##  1st Qu.:0.700          1st Qu.:2.000          1st Qu.:7.100
##  Median :1.000          Median :3.000          Median :7.200
##  Mean   :1.014          Mean   :2.534          Mean   :6.851
##  3rd Qu.:1.300          3rd Qu.:3.000          3rd Qu.:7.300
##  Max.   :2.300          Max.   :3.000          Max.   :7.500
##
##  StDev  :0.45           StDev  :0.65           StDev  :1.51
##  Skew   :0.38           Skew   :-1.7           Skew   :-4.31
##  ManufacturingProcess40 ManufacturingProcess41 ManufacturingProcess42
##  Min.   :0.00000        Min.   :0.00000        Min.   : 0.00
##  1st Qu.:0.00000        1st Qu.:0.00000        1st Qu.:11.40
##  Median :0.00000        Median :0.00000        Median :11.60
##  Mean   :0.01771        Mean   :0.02371        Mean   :11.21
##  3rd Qu.:0.00000        3rd Qu.:0.00000        3rd Qu.:11.70
##  Max.   :0.10000        Max.   :0.20000        Max.   :12.10
##  NA's   :1              NA's   :1
##  StDev  :0.04           StDev  :0.05           StDev  :1.94
##  Skew   :1.69           Skew   :2.19           Skew   :-5.5
##  ManufacturingProcess43 ManufacturingProcess44 ManufacturingProcess45
##  Min.   : 0.0000        Min.   :0.000          Min.   :0.000
##  1st Qu.: 0.6000        1st Qu.:1.800          1st Qu.:2.100
##  Median : 0.8000        Median :1.900          Median :2.200
##  Mean   : 0.9119        Mean   :1.805          Mean   :2.138
##  3rd Qu.: 1.0250        3rd Qu.:1.900          3rd Qu.:2.300
##  Max.   :11.0000        Max.   :2.100          Max.   :2.600
##
##  StDev  :0.87           StDev  :0.32           StDev  :0.41
##  Skew   :9.13           Skew   :-5.01          Skew   :-4.11
```

```r
# check rows for NAs
nTotalRows <- nrow(m_ChemicalManufacturingProcess)
nCompleteRows <- sum(completeRows <- complete.cases(m_ChemicalManufacturingProcess))
nRowsWithNA <- nTotalRows - nCompleteRows
print(paste("There are ", nCompleteRows, "Complete Rows and ", nRowsWithNA, "Rows with some NA value, o
```

```
## [1] "There are  152 Complete Rows and  24 Rows with some NA value, out of  176 Total Rows"
```

```r
# check columns for NAs
nTotalCols <- ncol(m_ChemicalManufacturingProcess)
colsWithNA <- apply(m_ChemicalManufacturingProcess,2,anyNA)
nColsWithNA <- sum(colsWithNA)
nCompleteCols <- nTotalCols - nColsWithNA
print(paste("There are ", nCompleteCols, "Complete Columns and ", nColsWithNA, "Columns with some NA va
```

```
## [1] "There are  30 Complete Columns and  28 Columns with some NA value, out of  58 Total Columns"
```

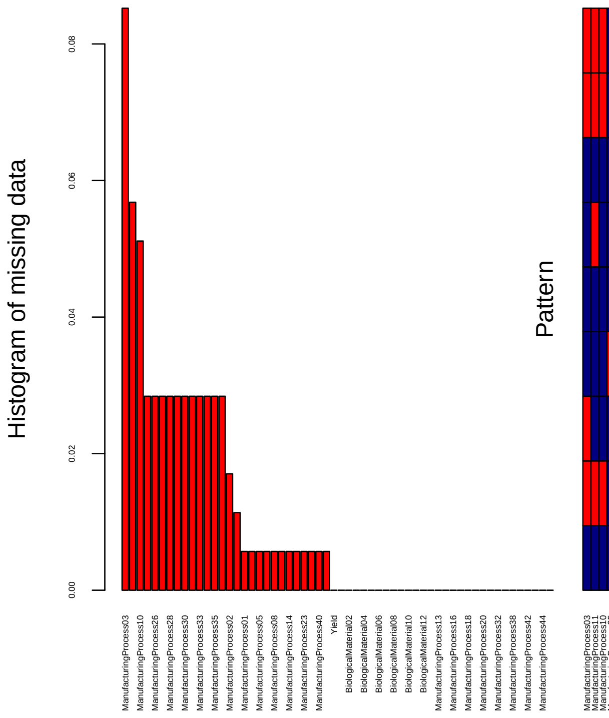
The matrix `ChemicalManufacturingProcess` contains the 57 predictors

- 12 describing the input biological material and
- 45 describing the process predictors) for the 176 manufacturing runs.
- `yield` contains the percent yield for each run.

(b) Imputation

A small percentage of cells in the predictor set contain missing values.

```r
#library(VIM)
ggr_plot <- aggr(
  origChemicalManufacturingProcess,
  col = c('navyblue', 'red'),
  numbers = TRUE,
  sortVars = TRUE,
  labels = names(origChemicalManufacturingProcess),
  cex.axis = .4,
  gap = 0.5,
  ylab = c("Histogram of missing data", "Pattern")
)
```



Visualize which columns have missing data using VIM::aggr :

```
##
##  Variables sorted by number of missings:
##               Variable       Count
##  ManufacturingProcess03 0.085227273
##  ManufacturingProcess11 0.056818182
##  ManufacturingProcess10 0.051136364
##  ManufacturingProcess25 0.028409091
##  ManufacturingProcess26 0.028409091
##  ManufacturingProcess27 0.028409091
##  ManufacturingProcess28 0.028409091
## ManufacturingProcess29 0.028409091
## ManufacturingProcess30 0.028409091
## ManufacturingProcess31 0.028409091
## ManufacturingProcess33 0.028409091
## ManufacturingProcess34 0.028409091
## ManufacturingProcess35 0.028409091
## ManufacturingProcess36 0.028409091
## ManufacturingProcess02 0.017045455
## ManufacturingProcess06 0.011363636
## ManufacturingProcess01 0.005681818
## ManufacturingProcess04 0.005681818
## ManufacturingProcess05 0.005681818
## ManufacturingProcess07 0.005681818
## ManufacturingProcess08 0.005681818
## ManufacturingProcess12 0.005681818
## ManufacturingProcess14 0.005681818
## ManufacturingProcess22 0.005681818
## ManufacturingProcess23 0.005681818
## ManufacturingProcess24 0.005681818
## ManufacturingProcess40 0.005681818
## ManufacturingProcess41 0.005681818
##                  Yield 0.000000000
##   BiologicalMaterial01 0.000000000
##   BiologicalMaterial02 0.000000000
##   BiologicalMaterial03 0.000000000
##   BiologicalMaterial04 0.000000000
##   BiologicalMaterial05 0.000000000
##   BiologicalMaterial06 0.000000000
##   BiologicalMaterial07 0.000000000
##   BiologicalMaterial08 0.000000000
##   BiologicalMaterial09 0.000000000
##   BiologicalMaterial10 0.000000000
##   BiologicalMaterial11 0.000000000
##   BiologicalMaterial12 0.000000000
## ManufacturingProcess09 0.000000000
## ManufacturingProcess13 0.000000000
## ManufacturingProcess15 0.000000000
## ManufacturingProcess16 0.000000000
## ManufacturingProcess17 0.000000000
## ManufacturingProcess18 0.000000000
## ManufacturingProcess19 0.000000000
## ManufacturingProcess20 0.000000000
## ManufacturingProcess21 0.000000000
## ManufacturingProcess32 0.000000000
## ManufacturingProcess37 0.000000000
## ManufacturingProcess38 0.000000000
## ManufacturingProcess39 0.000000000
## ManufacturingProcess42 0.000000000
## ManufacturingProcess43 0.000000000
## ManufacturingProcess44 0.000000000
## ManufacturingProcess45 0.000000000
```

Use an imputation function to fill in these missing values (e.g., see Sect. 3.8).

```r
#library(mice)
imputeChemicalManufacturingProcess <- mice(
  m_ChemicalManufacturingProcess,
  m = 2,
  maxit = 10,
  meth = 'pmm',
  seed = 500,
  print = F
)
```

Use MICE: "Multivariate Imputation by Chained Equations" to impute missing values

```
## Warning: Number of logged events: 540
```

```r
ChemicalManufacturingProcess <- complete(imputeChemicalManufacturingProcess)
m_ChemicalManufacturingProcess <- as.matrix(ChemicalManufacturingProcess)

### Any NA values?
anyNA(ChemicalManufacturingProcess)
```

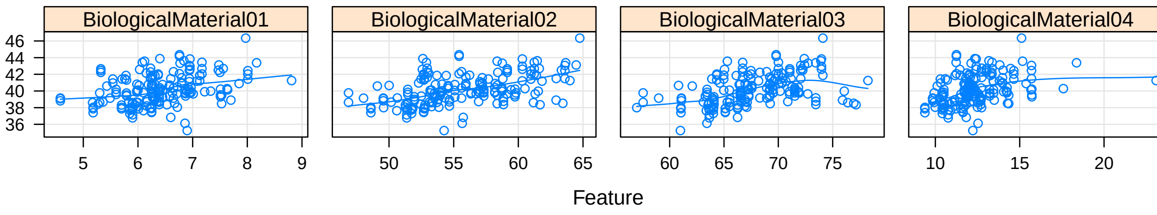
```
## [1] FALSE
```

```r
anyNA(m_ChemicalManufacturingProcess)
```

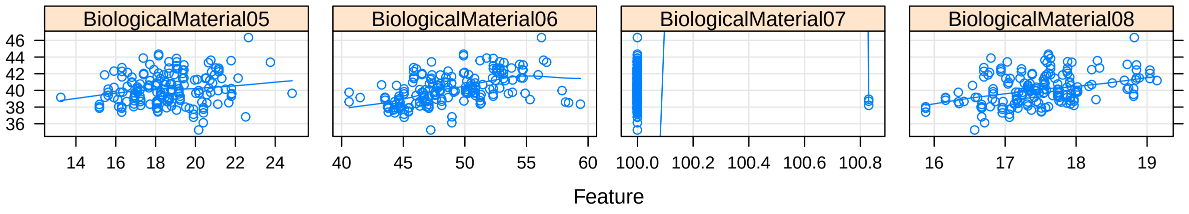
```
## [1] FALSE
```

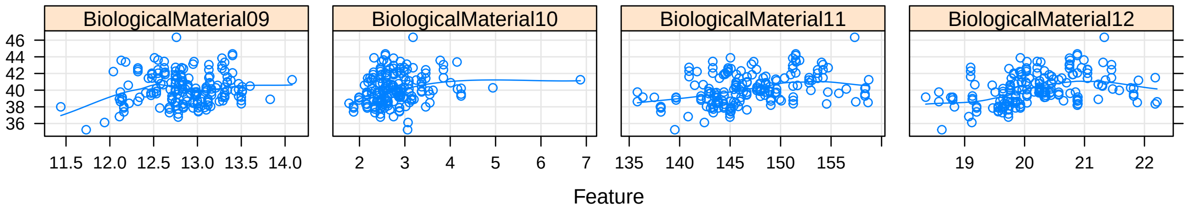
```r
##ggpairs(ChemicalManufacturingProcess[,1:10])
```

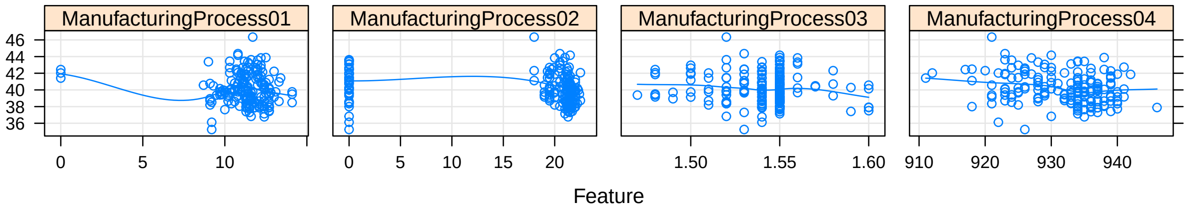
```r
for (low in seq(2,54,4)) {
  #print(paste("Range = ", low, " to ", low+3))
print(featurePlot(
  x = m_ChemicalManufacturingProcess[, low:(low+3)],
  y = m_ChemicalManufacturingProcess[, 1],
  between = list(x = 1, y = 1),
  type = c("g", "p", "smooth")
))
}
```

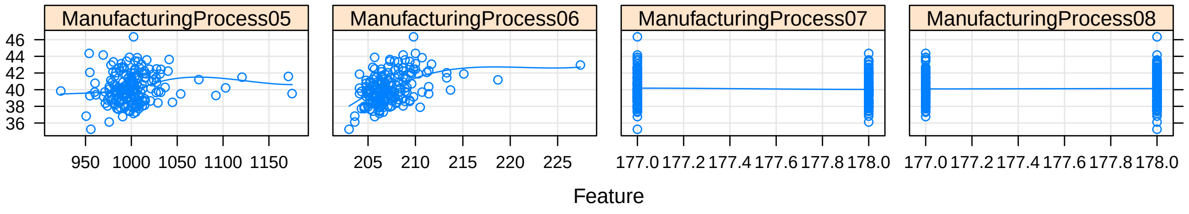


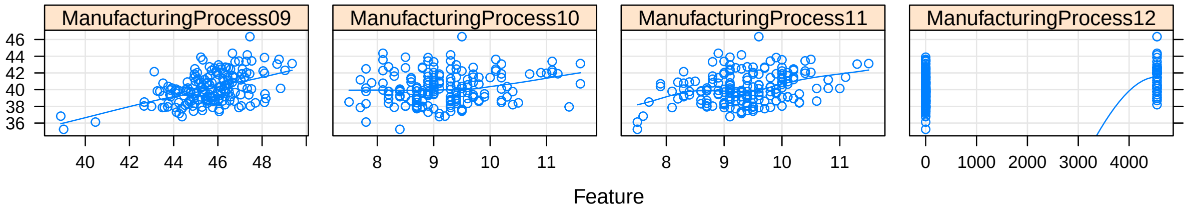
FeaturePlot Loop

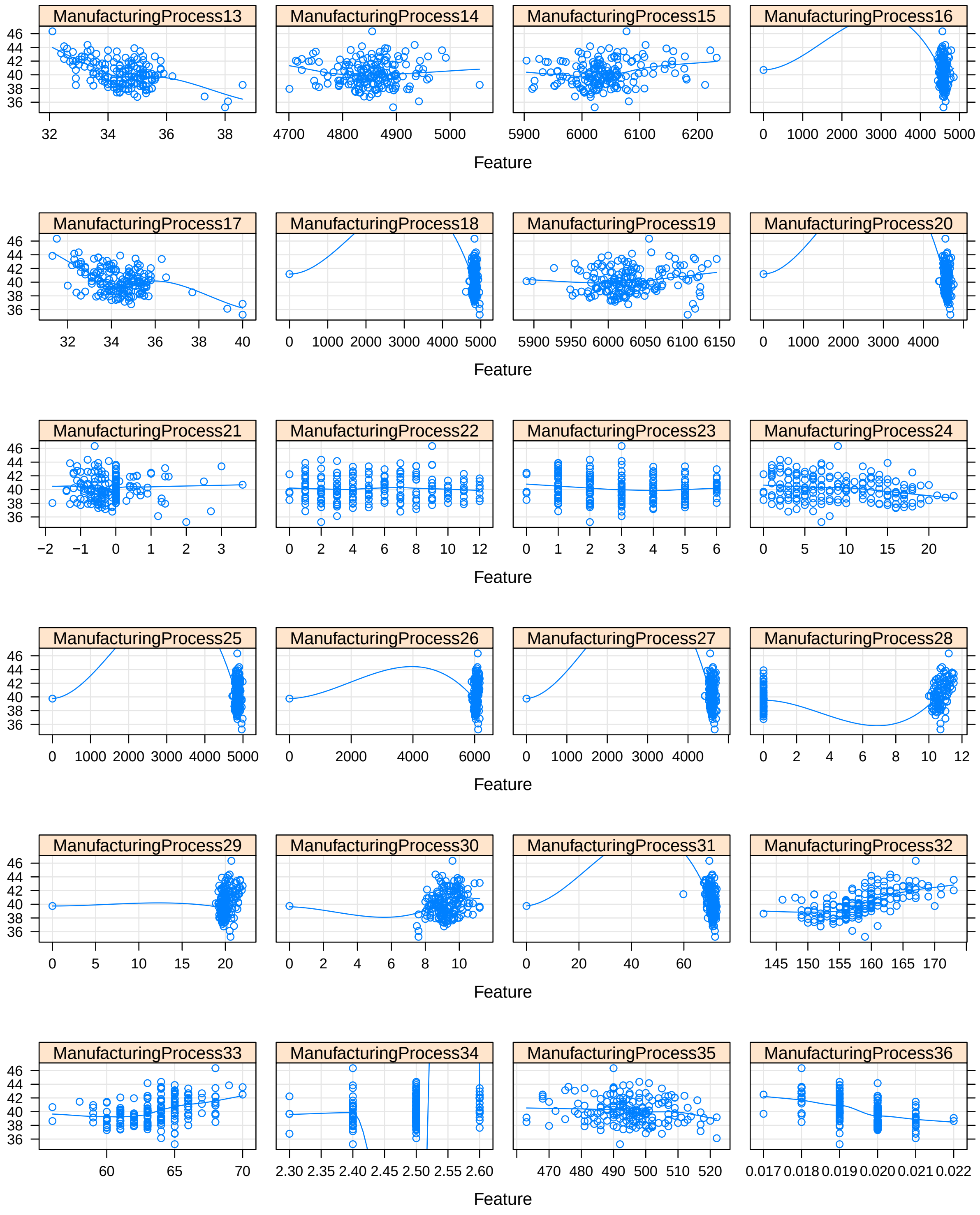


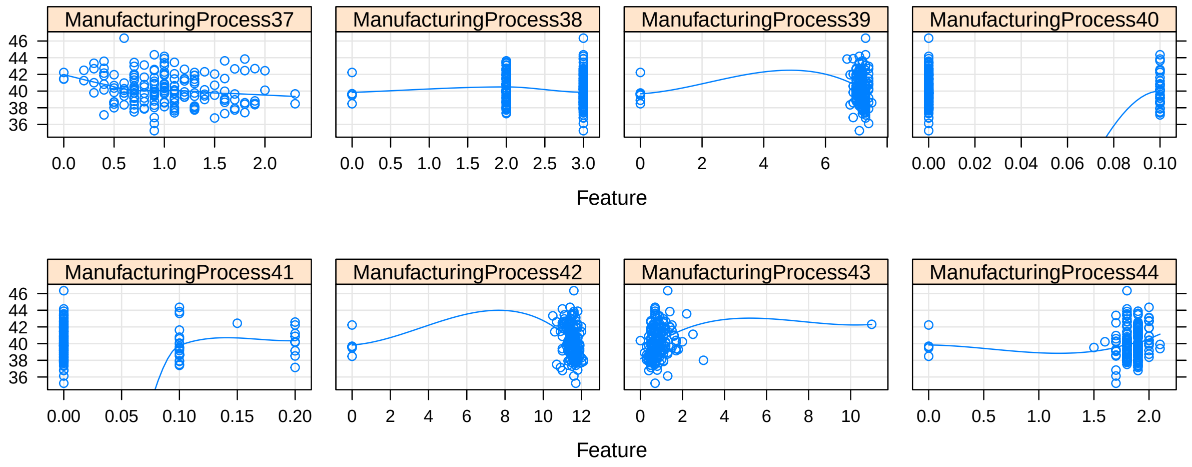









ManufacturingProcess13
ManufacturingProcess14
ManufacturingProcess15
ManufacturingProcess16
Feature
ManufacturingProcess17
ManufacturingProcess18
ManufacturingProcess19
ManufacturingProcess20
Feature
ManufacturingProcess21
ManufacturingProcess22
ManufacturingProcess23
ManufacturingProcess24
Feature
ManufacturingProcess25
ManufacturingProcess26
ManufacturingProcess27
ManufacturingProcess28
Feature
ManufacturingProcess29
ManufacturingProcess30
ManufacturingProcess31
ManufacturingProcess32
Feature
ManufacturingProcess33
ManufacturingProcess34
ManufacturingProcess35
ManufacturingProcess36
Feature



```r
### Above returns a data.frame
### Because all the data is numeric, we can change from data.frame to matrix
m_ChemicalManufacturingProcess <- as.matrix(ChemicalManufacturingProcess)
# Any NA values?
anyNA(m_ChemicalManufacturingProcess)
```

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

```r
#### Repeat summary on imputed matrix
rbind(summary(m_ChemicalManufacturingProcess),
      paste0("StDev  :",round(apply(X = m_ChemicalManufacturingProcess, MARGIN = 2, FUN = sd),2),"  "),
      paste0("Skew   :",round(skewness(m_ChemicalManufacturingProcess),2),"  ")) %>%
      as.table()
```

```
##      Yield       BiologicalMaterial01 BiologicalMaterial02 BiologicalMaterial03
##  Min.   :35.25   Min.   :4.580        Min.   :46.87        Min.   :56.97
##  1st Qu.:38.75   1st Qu.:5.978        1st Qu.:52.68        1st Qu.:64.98
##  Median :39.97   Median :6.305        Median :55.09        Median :67.22
##  Mean   :40.18   Mean   :6.411        Mean   :55.69        Mean   :67.70
##  3rd Qu.:41.48   3rd Qu.:6.870        3rd Qu.:58.74        3rd Qu.:70.43
##  Max.   :46.34   Max.   :8.810        Max.   :64.75        Max.   :78.25
##  StDev  :1.85    StDev  :0.71         StDev  :4.03         StDev  :4
##  Skew   :0.31    Skew   :0.28         Skew   :0.25         Skew   :0.03
##  BiologicalMaterial04 BiologicalMaterial05 BiologicalMaterial06
##  Min.   : 9.38        Min.   :13.24        Min.   :40.60
##  1st Qu.:11.24        1st Qu.:17.23        1st Qu.:46.05
##  Median :12.10        Median :18.49        Median :48.46
##  Mean   :12.35        Mean   :18.60        Mean   :48.91
##  3rd Qu.:13.22        3rd Qu.:19.90        3rd Qu.:51.34
##  Max.   :23.09        Max.   :24.85        Max.   :59.38
##  StDev  :1.77         StDev  :1.84         StDev  :3.75
##  Skew   :1.75         Skew   :0.31         Skew   :0.37
##  BiologicalMaterial07 BiologicalMaterial08 BiologicalMaterial09
##  Min.   :100.0        Min.   :15.88        Min.   :11.44
##  1st Qu.:100.0        1st Qu.:17.06        1st Qu.:12.60
##  Median :100.0        Median :17.51        Median :12.84
##  Mean   :100.0        Mean   :17.49        Mean   :12.85
##  3rd Qu.:100.0        3rd Qu.:17.88        3rd Qu.:13.13
##  Max.   :100.8        Max.   :19.14        Max.   :14.08
##  StDev  :0.11         StDev  :0.68         StDev  :0.42
##  Skew   :7.46         Skew   :0.22         Skew   :-0.27
##  BiologicalMaterial10 BiologicalMaterial11 BiologicalMaterial12
##  Min.   :1.770        Min.   :135.8        Min.   :18.35
##  1st Qu.:2.460        1st Qu.:143.8        1st Qu.:19.73
##  Median :2.710        Median :146.1        Median :20.12
##  Mean   :2.801        Mean   :147.0        Mean   :20.20
##  3rd Qu.:2.990        3rd Qu.:149.6        3rd Qu.:20.75
##  Max.   :6.870        Max.   :158.7        Max.   :22.21
##  StDev  :0.6          StDev  :4.82         StDev  :0.77
##  Skew   :2.42         Skew   :0.36         Skew   :0.31
##  ManufacturingProcess01 ManufacturingProcess02 ManufacturingProcess03
##  Min.   : 0.00          Min.   : 0.00          Min.   :1.47
##  1st Qu.:10.80          1st Qu.:19.23          1st Qu.:1.53
##  Median :11.40          Median :21.00          Median :1.54
##  Mean   :11.21          Mean   :16.64          Mean   :1.54
##  3rd Qu.:12.20          3rd Qu.:21.50          3rd Qu.:1.55
##  Max.   :14.10          Max.   :22.50          Max.   :1.60
##  StDev  :1.82           StDev  :8.51           StDev  :0.02
##  Skew   :-3.96          Skew   :-1.43          Skew   :-0.4
##  ManufacturingProcess04 ManufacturingProcess05 ManufacturingProcess06
##  Min.   :911.0          Min.   : 923.0         Min.   :203.0
##  1st Qu.:927.8          1st Qu.: 986.8         1st Qu.:205.7
##  Median :934.0          Median : 999.0         Median :206.8
##  Mean   :931.8          Mean   :1001.6         Mean   :207.4
##  3rd Qu.:936.0          3rd Qu.:1008.7         3rd Qu.:208.7
##  Max.   :946.0          Max.   :1175.3         Max.   :227.4
##  StDev  :6.34           StDev  :30.45          StDev  :2.7
##  Skew   :-0.7           Skew   :2.62           Skew   :3.07
##  ManufacturingProcess07 ManufacturingProcess08 ManufacturingProcess09
##  Min.   :177.0          Min.   :177.0          Min.   :38.89
##  1st Qu.:177.0          1st Qu.:177.0          1st Qu.:44.89
##  Median :177.0          Median :178.0          Median :45.73
##  Mean   :177.5          Mean   :177.6          Mean   :45.66
##  3rd Qu.:178.0          3rd Qu.:178.0          3rd Qu.:46.52
##  Max.   :178.0          Max.   :178.0          Max.   :49.36
##  StDev  :0.5            StDev  :0.5            StDev  :1.55
##  Skew   :0.07           Skew   :-0.23          Skew   :-0.95
##  ManufacturingProcess10 ManufacturingProcess11 ManufacturingProcess12
##  Min.   : 7.500         Min.   : 7.500         Min.   :   0.0
##  1st Qu.: 8.700         1st Qu.: 9.000         1st Qu.:   0.0
##  Median : 9.050         Median : 9.400         Median :   0.0
##  Mean   : 9.170         Mean   : 9.384         Mean   : 852.9
##  3rd Qu.: 9.525         3rd Qu.: 9.900         3rd Qu.:   0.0
##  Max.   :11.600         Max.   :11.500         Max.   :4549.0
##  StDev  :0.79           StDev  :0.72           StDev  :1780.6
##  Skew   :0.68           Skew   :-0.02          Skew   :1.6
##  ManufacturingProcess13 ManufacturingProcess14 ManufacturingProcess15
##  Min.   :32.10          Min.   :4701           Min.   :5904
##  1st Qu.:33.90          1st Qu.:4827           1st Qu.:6010
##  Median :34.60          Median :4856           Median :6032
##  Mean   :34.51          Mean   :4853           Mean   :6039
##  3rd Qu.:35.20          3rd Qu.:4882           3rd Qu.:6061
##  Max.   :38.60          Max.   :5055           Max.   :6233
##  StDev  :1.02           StDev  :55.24          StDev  :58.31
##  Skew   :0.48           Skew   :-0.04          Skew   :0.68
##  ManufacturingProcess16 ManufacturingProcess17 ManufacturingProcess18
##  Min.   :   0           Min.   :31.30          Min.   :   0
##  1st Qu.:4561           1st Qu.:33.50          1st Qu.:4813
##  Median :4588           Median :34.40          Median :4835
##  Mean   :4566           Mean   :34.34          Mean   :4810
##  3rd Qu.:4619           3rd Qu.:35.10          3rd Qu.:4862
##  Max.   :4852           Max.   :40.00          Max.   :4971
##  StDev  :351.7          StDev  :1.25           StDev  :367.48
##  Skew   :-12.53         Skew   :1.17           Skew   :-12.85
##  ManufacturingProcess19 ManufacturingProcess20 ManufacturingProcess21
##  Min.   :5890           Min.   :   0           Min.   :-1.8000
##  1st Qu.:6001           1st Qu.:4553           1st Qu.:-0.6000
##  Median :6022           Median :4582           Median :-0.3000
##  Mean   :6028           Mean   :4556           Mean   :-0.1642
##  3rd Qu.:6050           3rd Qu.:4610           3rd Qu.: 0.0000
##  Max.   :6146           Max.   :4759           Max.   : 3.6000
##  StDev  :45.58          StDev  :349.01         StDev  :0.78
##  Skew   :0.3            Skew   :-12.75         Skew   :1.74
##  ManufacturingProcess22 ManufacturingProcess23 ManufacturingProcess24
##  Min.   : 0.000         Min.   :0.000          Min.   : 0.000
##  1st Qu.: 3.000         1st Qu.:2.000          1st Qu.: 4.000
##  Median : 5.000         Median :3.000          Median : 8.000
##  Mean   : 5.415         Mean   :3.006          Mean   : 8.841
##  3rd Qu.: 8.000         3rd Qu.:4.000          3rd Qu.:14.000
##  Max.   :12.000         Max.   :6.000          Max.   :23.000
##  StDev  :3.32           StDev  :1.66           StDev  :5.78
##  Skew   :0.31           Skew   :0.21           Skew   :0.36
##  ManufacturingProcess25 ManufacturingProcess26 ManufacturingProcess27
##  Min.   :   0           Min.   :   0           Min.   :   0
##  1st Qu.:4834           1st Qu.:6019           1st Qu.:4563
##  Median :4856           Median :6045           Median :4588
##  Mean   :4832           Mean   :6014           Mean   :4564
##  3rd Qu.:4882           3rd Qu.:6069           3rd Qu.:4610
##  Max.   :4990           Max.   :6161           Max.   :4710
##  StDev  :368.73         StDev  :458.37         StDev  :349.08
##  Skew   :-12.89         Skew   :-12.94         Skew   :-12.8
##  ManufacturingProcess28 ManufacturingProcess29 ManufacturingProcess30
##  Min.   : 0.000         Min.   : 0.0           Min.   : 0.000
##  1st Qu.: 0.000         1st Qu.:19.7           1st Qu.: 8.800
##  Median :10.400         Median :19.9           Median : 9.200
##  Mean   : 6.405         Mean   :20.0           Mean   : 9.209
##  3rd Qu.:10.700         3rd Qu.:20.4           3rd Qu.: 9.700
##  Max.   :11.500         Max.   :22.0           Max.   :11.200
##  StDev  :5.29           StDev  :1.64           StDev  :1
##  Skew   :-0.39          Skew   :-10.24         Skew   :-4.29
##  ManufacturingProcess31 ManufacturingProcess32 ManufacturingProcess33
##  Min.   : 0.00          Min.   :143.0          Min.   :56.00
##  1st Qu.:70.10          1st Qu.:155.0          1st Qu.:62.00
##  Median :70.80          Median :158.0          Median :64.00
##  Mean   :70.24          Mean   :158.5          Mean   :63.66
##  3rd Qu.:71.40          3rd Qu.:162.0          3rd Qu.:65.00
##  Max.   :72.50          Max.   :173.0          Max.   :70.00
##  StDev  :5.49           StDev  :5.4            StDev  :2.54
##  Skew   :-12.07         Skew   :0.21           Skew   :-0.13
##  ManufacturingProcess34 ManufacturingProcess35 ManufacturingProcess36
##  Min.   :2.300          Min.   :463            Min.   :0.01700
##  1st Qu.:2.500          1st Qu.:490            1st Qu.:0.01900
##  Median :2.500          Median :495            Median :0.01900
##  Mean   :2.489          Mean   :495            Mean   :0.01953
##  3rd Qu.:2.500          3rd Qu.:501            3rd Qu.:0.02000
##  Max.   :2.600          Max.   :522            Max.   :0.02200
##  StDev  :0.06           StDev  :11.32          StDev  :0
##  Skew   :-0.59          Skew   :-0.24          Skew   :0.05
##  ManufacturingProcess37 ManufacturingProcess38 ManufacturingProcess39
##  Min.   :0.000          Min.   :0.000          Min.   :0.000
##  1st Qu.:0.700          1st Qu.:2.000          1st Qu.:7.100
##  Median :1.000          Median :3.000          Median :7.200
##  Mean   :1.014          Mean   :2.534          Mean   :6.851
##  3rd Qu.:1.300          3rd Qu.:3.000          3rd Qu.:7.300
##  Max.   :2.300          Max.   :3.000          Max.   :7.500
##  StDev  :0.45           StDev  :0.65           StDev  :1.51
##  Skew   :0.38           Skew   :-1.7           Skew   :-4.31
##  ManufacturingProcess40 ManufacturingProcess41 ManufacturingProcess42
##  Min.   :0.00000        Min.   :0.00000        Min.   : 0.00
##  1st Qu.:0.00000        1st Qu.:0.00000        1st Qu.:11.40
##  Median :0.00000        Median :0.00000        Median :11.60
##  Mean   :0.01761        Mean   :0.02358        Mean   :11.21
##  3rd Qu.:0.00000        3rd Qu.:0.00000        3rd Qu.:11.70
##  Max.   :0.10000        Max.   :0.20000        Max.   :12.10
##  StDev  :0.04           StDev  :0.05           StDev  :1.94
##  Skew   :1.7            Skew   :2.2            Skew   :-5.5
##  ManufacturingProcess43 ManufacturingProcess44 ManufacturingProcess45
##  Min.   : 0.0000        Min.   :0.000          Min.   :0.000
##  1st Qu.: 0.6000        1st Qu.:1.800          1st Qu.:2.100
##  Median : 0.8000        Median :1.900          Median :2.200
##  Mean   : 0.9119        Mean   :1.805          Mean   :2.138
##  3rd Qu.: 1.0250        3rd Qu.:1.900          3rd Qu.:2.300
##  Max.   :11.0000        Max.   :2.100          Max.   :2.600
##  StDev  :0.87           StDev  :0.32           StDev  :0.41
##  Skew   :9.13           Skew   :-5.01          Skew   :-4.11
```

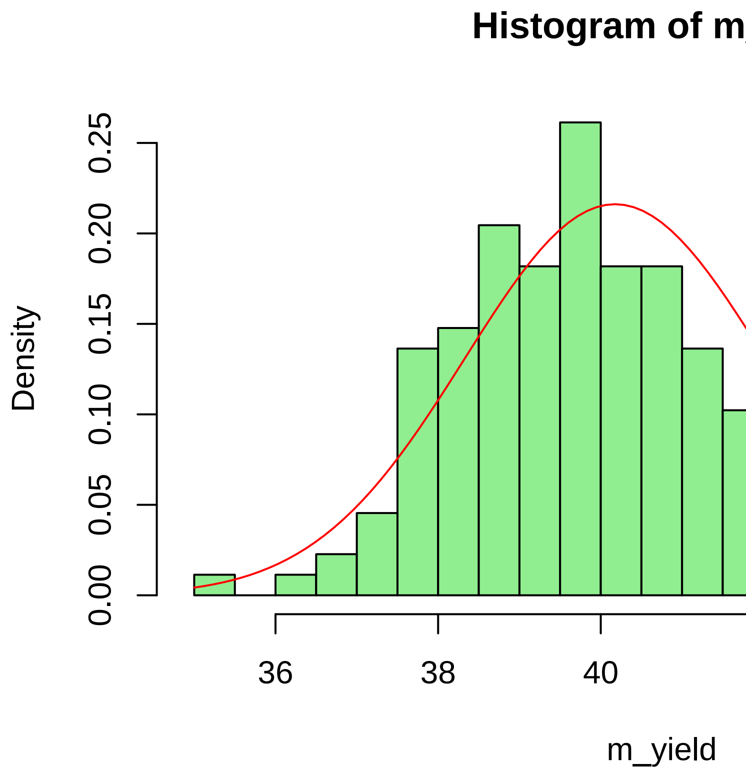
```r
yield = ChemicalManufacturingProcess[,1]
predictors = ChemicalManufacturingProcess[,2:nTotalCols]

#ggpairs(predictors)

#### some code expects yield and predictors to be matrix, not array or dataframe
#### All values are numeric, so we can do this
m_yield <- as.matrix(yield)
m_predictors <- as.matrix(predictors)
```

```
nSamples = dim(m_predictors)[1]
nFeatures = dim(m_predictors)[2]

print(paste("Total number of cases is",nSamples,"; total number of features is", nFeatures))
```

Separate out the target variable ("Yield") from the predictors

```
## [1] "Total number of cases is 176 ; total number of features is 57"
```

```
# histogram of yield
hist(m_yield,prob=T,breaks=20,col="lightgreen")
curve(dnorm(x, mean = mean(m_yield), sd = sd(m_yield)), col="red", add=TRUE)
```



Histogram of yield, with normal density curve(red)

```
#library(olsrr)
ols_test_normality(m_yield)
```

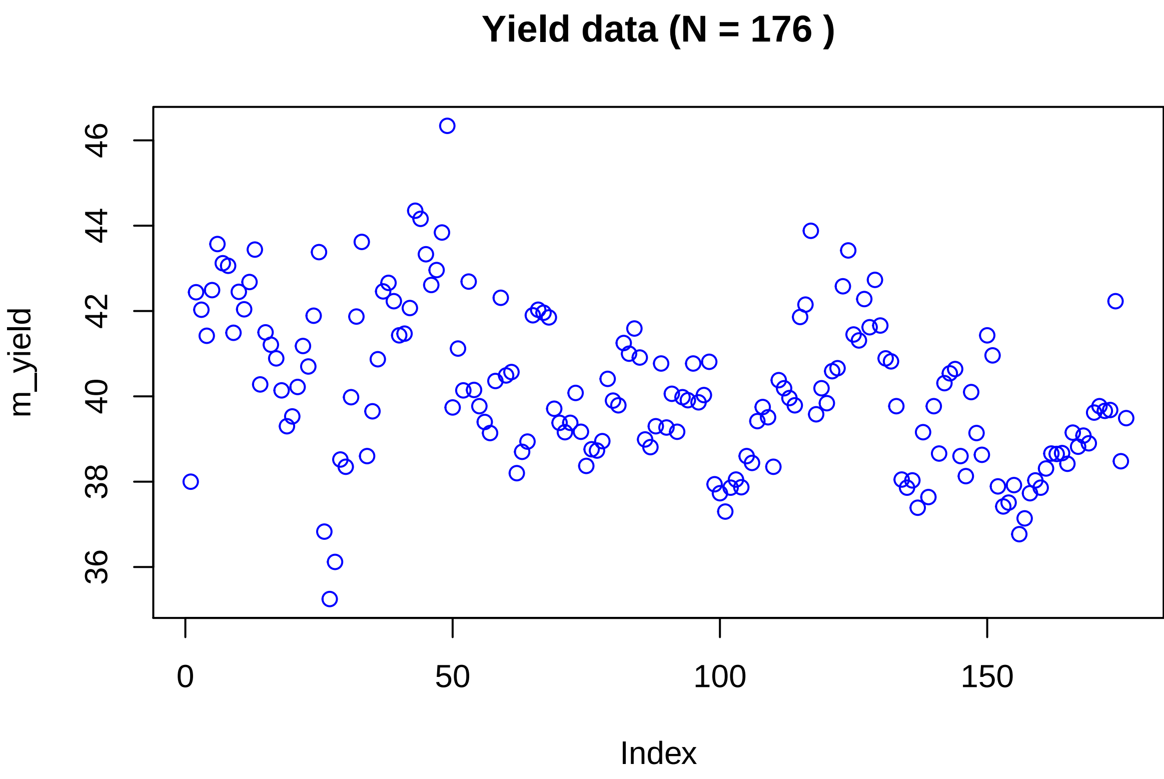
Tests for normality

```
## Warning in ks.test(y, "pnorm", mean(y), sd(y)): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
## -------------------------------------------------
##        Test                Statistic       pvalue
## -------------------------------------------------
## Shapiro-Wilk                  0.9885         0.1647
## Kolmogorov-Smirnov            0.0596         0.5596
## Cramer-von Mises             58.6667         0.0000
## Anderson-Darling              0.7048         0.0647
## -------------------------------------------------
```

Because 3 of 4 normality tests are passed, there will be no need to transform the yield variable.

```
mainlabel=paste("Yield data (N =",nSamples,")")
plot(m_yield,main=mainlabel,col="blue")
```



scatterplot of yield

Check for high correlations between columns in the ChemicalManufacturingProcess data set:

```
### be sure to specify corrplot::corrplot because the namespace may be masked by pls::corrplot
#library(corrplot)
correl5 <- cor(m_ChemicalManufacturingProcess)

# determinant is (barely) non-zero
print(paste("Determinant: ", det(correl5)))
```

Correlation grid #1

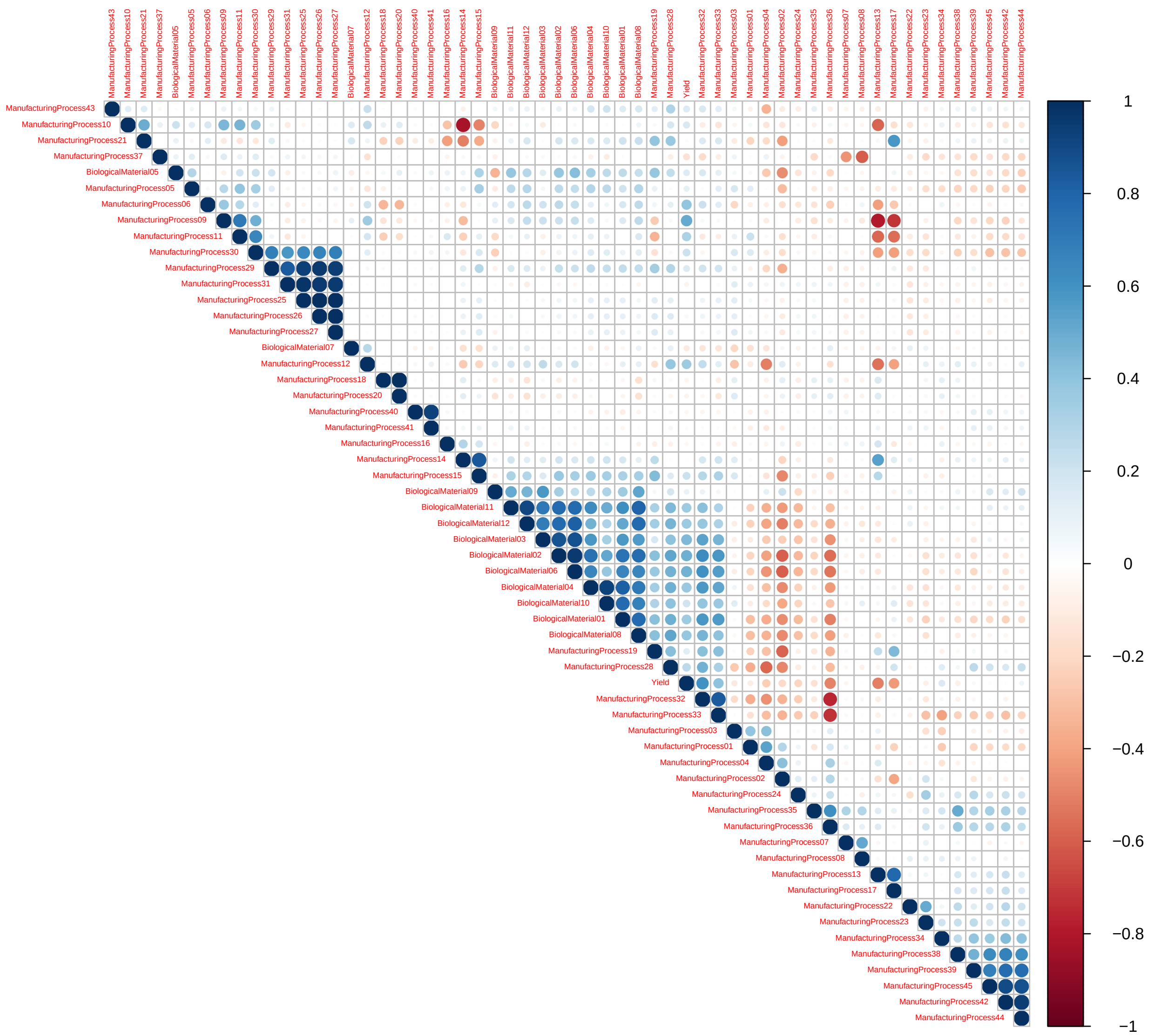
```
## [1] "Determinant:  -0.000000000000000000000000000000000000000000000000000000733930215789486"
```

```
# some columns are very similar to other columns.
maxcor5 <- round(max(correl5-diag(1,ncol(correl5),ncol(correl5))),5)
mincor5 <- round(min(correl5-diag(1,ncol(correl5),ncol(correl5))),5)
print(paste("Range of off-diag correlations: ",
            paste0("[",mincor5,",",maxcor5,"]"), "on",ncol(correl5),"columns"))
```

```
## [1] "Range of off-diag correlations:  [-0.82847,0.99444] on 58 columns"
```

```
corrplot::corrplot(
  correl5,
  method = "circle",
  type = "upper",
  order = "hclust",
  tl.cex = 0.4,
  main = paste("\nClustered correlations of ChemicalManufacturingProcess (ncol=",
               ncol(correl5),")")
)
```

Clustered correlations of ChemicalManufacturingProcess (ncol= 58)



```r
correl6 <- cor(m_predictors)

# determinant is (barely) nonzero
print(paste("Determinant: ", det(correl6)))
```

Select high-correlation columns to be dropped

```
## [1] "Determinant:  -0.0000000000000000000000000000000000000000000000000000027892858497667l"
```

Table 2: Columns to be dropped

x
BiologicalMaterial02
BiologicalMaterial04
BiologicalMaterial12
ManufacturingProcess18
ManufacturingProcess26
ManufacturingProcess27
ManufacturingProcess29
ManufacturingProcess31
ManufacturingProcess40
ManufacturingProcess42

```
# some columns are very similar to other columns.
maxcor6 <- round(max(correl6-diag(1,ncol(correl6),ncol(correl6))),5)
mincor6 <- round(min(correl6-diag(1,ncol(correl6),ncol(correl6))),5)
print(paste("Range of off-diag predictor correlations: ",
            paste0("[",mincor6,",",maxcor6,"]"), "on",ncol(correl6),"columns"))
```

Range of correlations

```
## [1] "Range of off-diag predictor correlations:  [-0.82847,0.99444] on 57 columns"
```

```
cutoff = 0.9
highcorrcols <- findCorrelation(correl6,cutoff = cutoff)
num_highcorrcols <- length(highcorrcols)
print(paste("Quantity of columns which have correlation > ", cutoff, ":",
            num_highcorrcols, "out of",ncol(correl6)))
```

eliminate predictor columns which are highly correlated to other predictor columns

```
## [1] "Quantity of columns which have correlation >  0.9 : 10 out of 57"
```

```
colnames(m_predictors)[highcorrcols] %>% sort()  %>%
  kable(caption = "Columns to be dropped") %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

Names of columns to be dropped:

```
# drop columns which have high correlation to some other column
m_predictors2 <- m_predictors[,-highcorrcols]
dim(m_predictors2)
```

Drop above predictors

```
## [1] 176  47
```

```
print(paste("Remaining number of predictor columns: ", ncol(m_predictors2)))
```

```
## [1] "Remaining number of predictor columns:  47"
```

```
correl7 <- cor(m_predictors2)
# determinant is (barely) nonzero
print(paste("Determinant: ", det(correl7)))
```

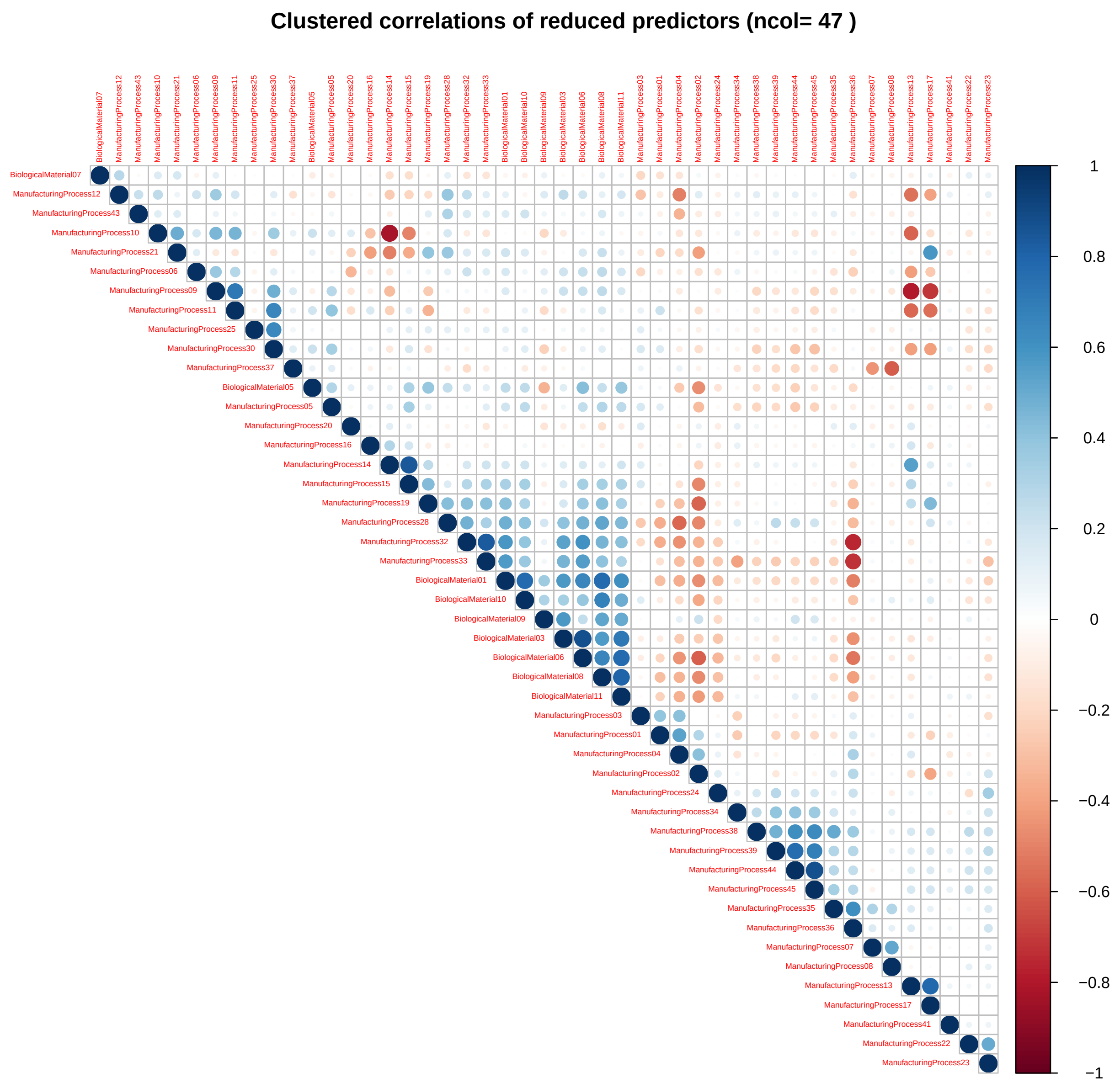
Correlation Plot of reduced predictors

```
## [1] "Determinant:  0.000000000000000000000000000000000000672927803149675"
```

```
# some columns are very similar to other columns.
maxcor7 <- round(max(correl7-diag(1,ncol(correl7),ncol(correl7))),5)
mincor7 <- round(min(correl7-diag(1,ncol(correl7),ncol(correl7))),5)
print(paste("Range of off-diag correlations: ",
            paste0("[",mincor7,",",maxcor7,"]"), "on",ncol(correl7),"columns"))
```

```
## [1] "Range of off-diag correlations:  [-0.82847,0.87729] on 47 columns"
```

```
corrplot::corrplot(
  correl7,
  method = "circle",
  type = "upper",
  order = "hclust",
  tl.cex = 0.4,
  main = paste("\nClustered correlations of reduced predictors (ncol=",
               ncol(correl7),")")
)
```



```r
predictors_nearZeroVarCols <- nearZeroVar(m_predictors2)
predictors_nTotalCols <- ncol(m_predictors2)
predictors_nDropCols <- length(predictors_nTotalCols)

print(paste("Number of NearZeroVar columns to be dropped:",
            predictors_nDropCols, "out of", predictors_nTotalCols))
```

Remove Near-Zero Variance predictors

```
## [1] "Number of NearZeroVar columns to be dropped: 1 out of 47"
```

Table 3: Columns to be dropped

x
BiologicalMaterial07

```
colnames(m_predictors2)[predictors_nearZeroVarCols]  %>%
  kable(caption = "Columns to be dropped") %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

```
m_predictors3 <- m_predictors2[,-predictors_nearZeroVarCols]
predictors_nFiltered <- ncol(m_predictors3)
dim(m_predictors3)
```

```
## [1] 176  46
```

(c) Split the data

```
set.seed(12345)
KJ63trainRow <- createDataPartition(m_yield, p=0.8, list=FALSE)


KJ63predictors.train     <- m_predictors3[KJ63trainRow, ]
KJ63yield.train          <- m_yield[KJ63trainRow, ]
KJ63predictors.test      <- m_predictors3[-KJ63trainRow, ]
KJ63yield.test           <- m_yield[-KJ63trainRow, ]
```

into a training and a test set,

```
preProc <- preProcess(KJ63predictors.train,
                      method=c("YeoJohnson","center","scale","knnImpute"))
preProcKJ63predictors.train    <- predict(preProc,KJ63predictors.train)
preProcKJ63predictors.test     <- predict(preProc,KJ63predictors.test)
```

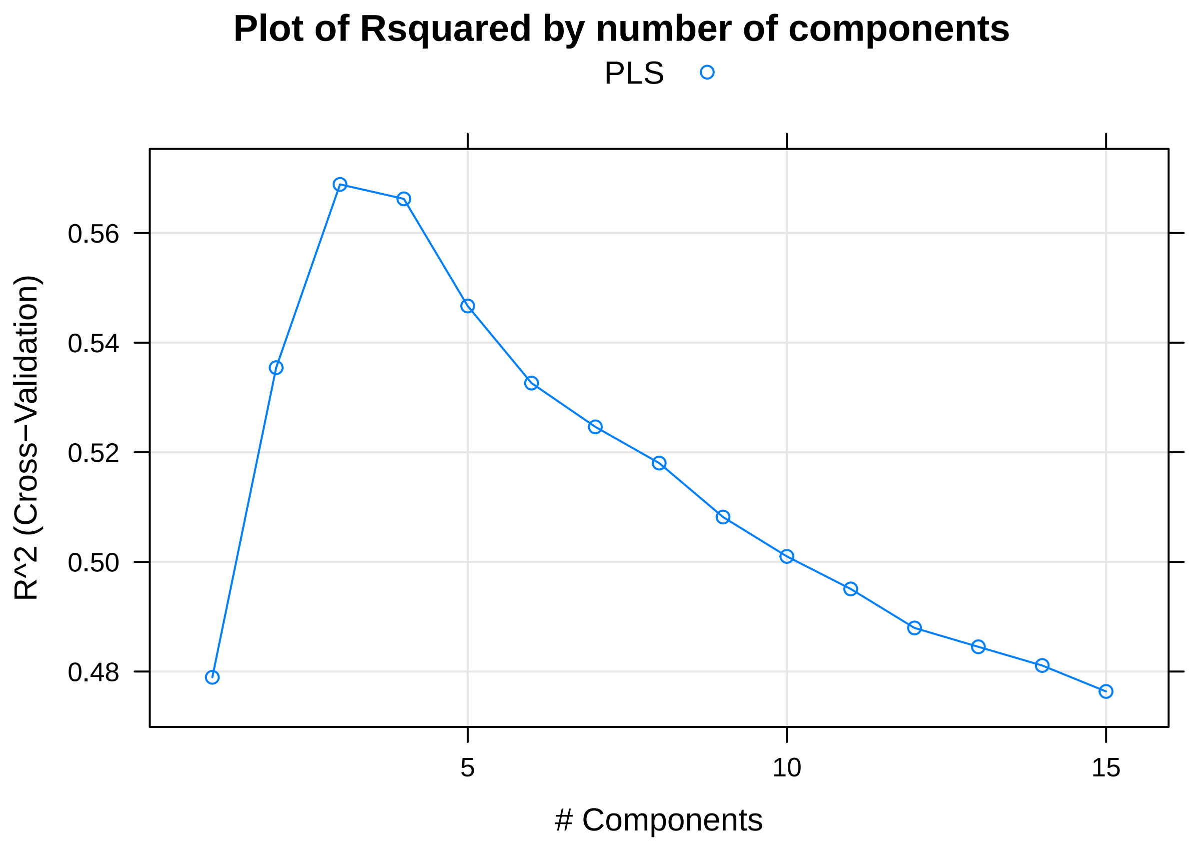
pre-process the data,

```
set.seed(517)
##ctrl <- trainControl(method = "cv")
KJ63ctrl <- trainControl(method = "boot", number = 25)
plsTune <- train(x          = preProcKJ63predictors.train,
                 y          = KJ63yield.train,
                 method     = "pls",
                 metric     = 'Rsquared',
                 tuneLength = 15,
                 #preProcess = c("YeoJohnson","center","scale"),
                 trControl  = KJ63ctrl)
plsTune
```

tune a model of your choice from this chapter.

```
## Partial Least Squares
##
## 144 samples
##  46 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 144, 144, 144, 144, 144, 144, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared   MAE
##    1      1.351370  0.4789476  1.088546
##    2      1.281916  0.5354338  1.033149
##    3      1.248938  0.5688726  1.007485
##    4      1.266925  0.5662339  1.013610
##    5      1.317003  0.5466883  1.044936
##    6      1.353998  0.5326223  1.071998
##    7      1.376628  0.5246330  1.096660
##    8      1.394664  0.5180284  1.112775
##    9      1.427052  0.5081854  1.140822
##   10      1.448483  0.5010053  1.154095
##   11      1.468570  0.4950663  1.166082
##   12      1.496297  0.4879506  1.182002
##   13      1.520226  0.4845110  1.191137
##   14      1.544901  0.4810994  1.199692
##   15      1.568391  0.4763670  1.209632
##
## Rsquared was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 3.
```

```r
plsResamples <- plsTune$results
plsResamples$Model <- "PLS"

xyplot(Rsquared ~ ncomp,
       data = plsResamples,
       #aspect = 1,
       xlab = "# Components",
       ylab = "R^2 (Cross-Validation)",
       auto.key = list(),
       groups = Model,
       type = c("o", "g"),
       main="Plot of Rsquared by number of components")
```



```
plsTune$bestTune$ncomp
```

What is the optimal value of the performance metric?

```
## [1] 3
```

```
plsTune$results[rownames(plsTune$bestTune),]
```

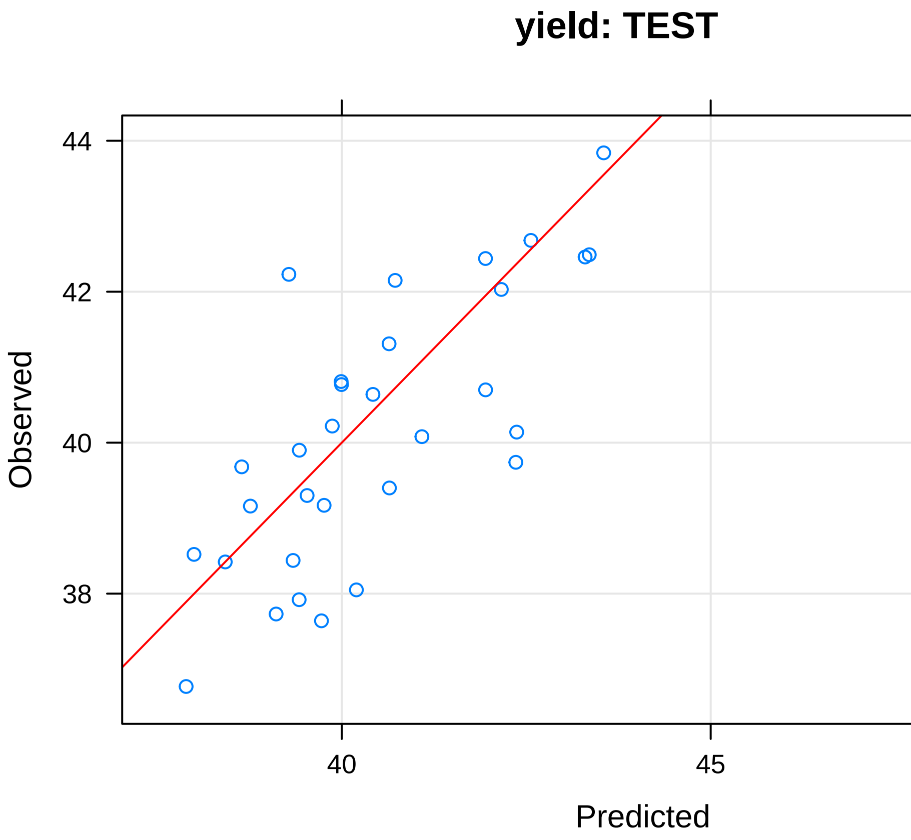
```
##   ncomp     RMSE  Rsquared      MAE     RMSESD RsquaredSD      MAESD
## 3     3 1.248938 0.5688726 1.007485 0.08554805 0.06793027 0.08187371
```

The optimal value occurs when the number of components is 3, where $R^2 = 0.5688726$.

(d) Predict the response for the test set.

```
pls_test_predictions <- predict(object = plsTune, newdata = preProcKJ63predictors.test    )
KJ63pls_test_stats <- postResample(pred = pls_test_predictions, obs = KJ63yield.test)
```

```
xyplot(KJ63yield.test ~ pls_test_predictions,
## plot the points (type = 'p') and a background grid ('g')
  type = c("p", "g"),
  xlab = "Predicted",
  ylab = "Observed",
  main="yield: TEST",
panel = function(x,y, ...){
  panel.xyplot(x,y, ...)
  panel.abline(a=0,b=1,col="red")
  }
)
```



Plot test data - predicted vs. observed

```
(KJ63pls_test_stats <- rbind(KJ63pls_test_stats)) %>%
  kable() %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

	RMSE	Rsquared	MAE
KJ63pls_test_stats	2.224497	0.2241122	1.287331

What is the value of the performance metric For the resampled training data, $R^2 = 0.2241122$ and RMSE $= 2.2244973$.

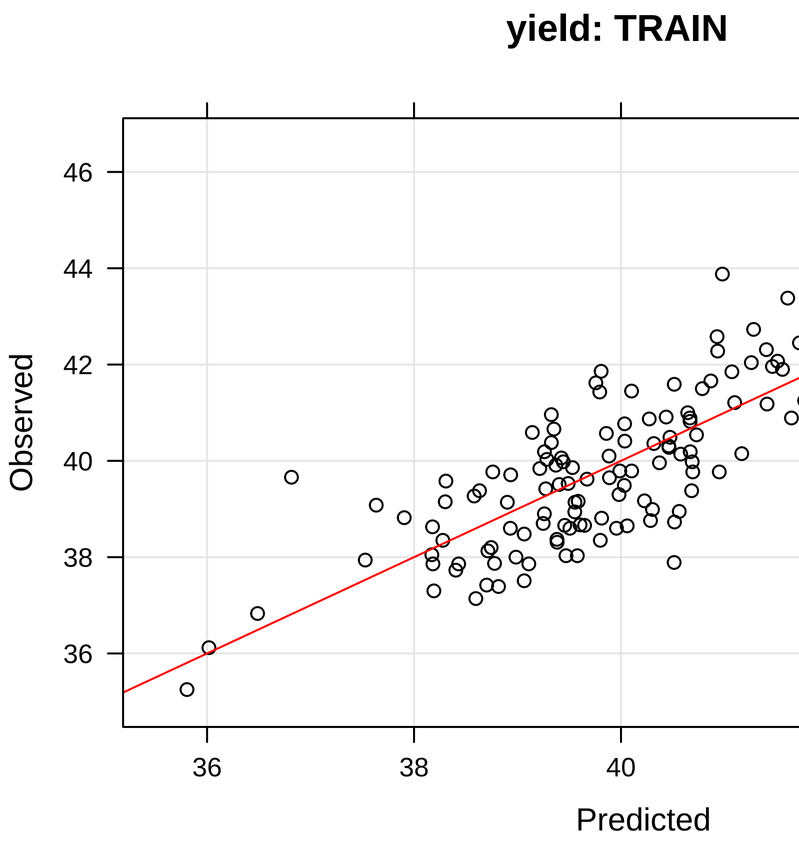
```
pls_train_predictions <- predict(object = plsTune) # , newdata = preProcKJ63predictors.train   )
KJ63pls_train_stats <- postResample(pred = pls_train_predictions, obs = KJ63yield.train)
(KJ63pls_train_stats <- rbind(KJ63pls_train_stats)) %>%
  kable() %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

	RMSE	Rsquared	MAE
KJ63pls_train_stats	1.037505	0.6900329	0.8429612

and how does this compare with the resampled performance metric on the training set? For the resampled training data, $R^2 = 0.6900329$ and RMSE $= 1.0375052$.

These are much better than the results on the test set, which suggests that there may be an overfitting problem,or that this model may not be the best choice.

```
xyplot(KJ63yield.train ~ pls_train_predictions,
## plot the points (type = 'p') and a background grid ('g')
  type = c("p", "g"),
  xlab = "Predicted",
  ylab = "Observed",
  main="yield: TRAIN",
  col="black",
panel = function(x,y, ...){
  panel.xyplot(x,y, ...)
  panel.abline(a=0,b=1,col="red")
  }
)
```



Plot TRAINING data - predicted vs. observed

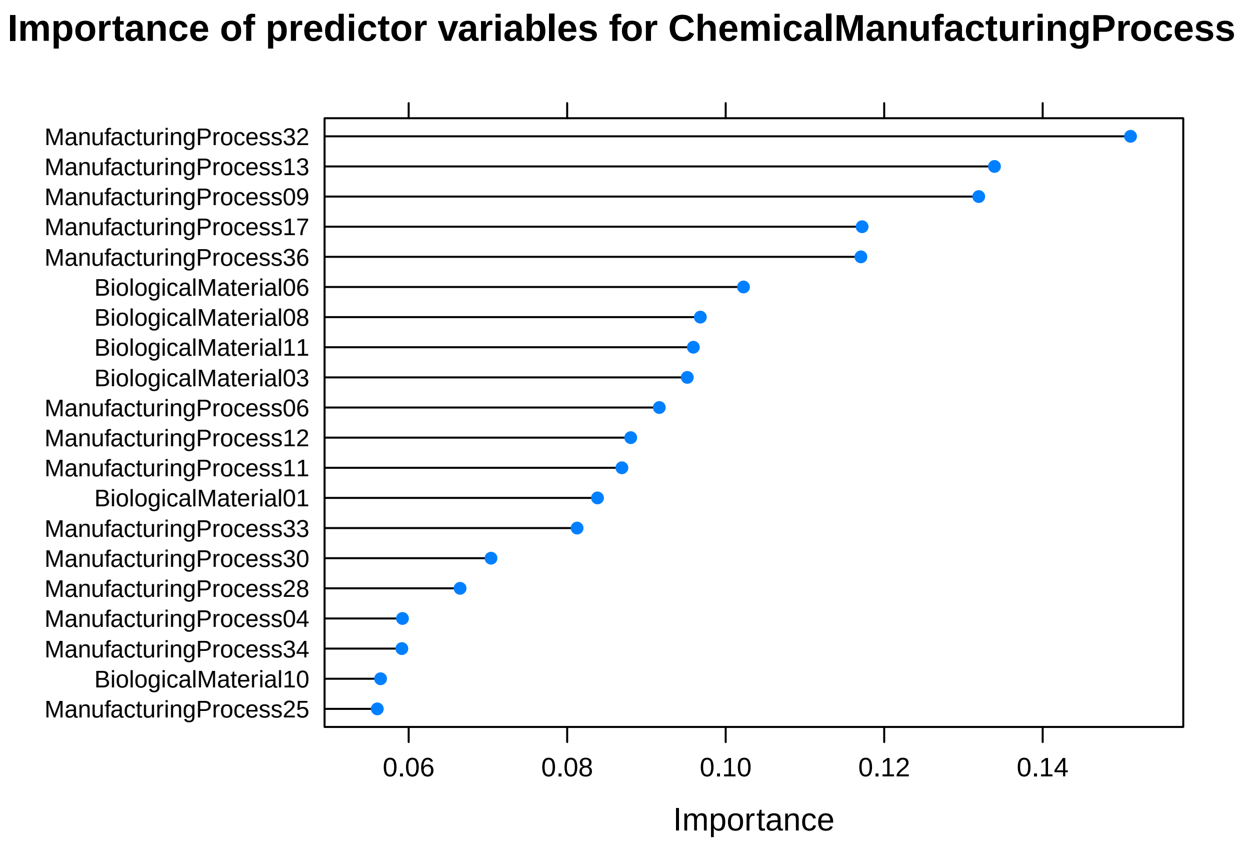
(e) Importance

```r
plsImp <- varImp(plsTune, scale = FALSE)
(plsImp)
```

Which predictors are most important in the model you have trained?

```
## pls variable importance
##
##   only 20 most important variables shown (out of 46)
##
##                        Overall
## ManufacturingProcess32 0.15109
## ManufacturingProcess13 0.13392
## ManufacturingProcess09 0.13193
## ManufacturingProcess17 0.11721
## ManufacturingProcess36 0.11706
## BiologicalMaterial06   0.10225
## BiologicalMaterial08   0.09680
## BiologicalMaterial11   0.09592
## BiologicalMaterial03   0.09516
## ManufacturingProcess06 0.09163
## ManufacturingProcess12 0.08802
## ManufacturingProcess11 0.08690
## BiologicalMaterial01   0.08383
## ManufacturingProcess33 0.08125
## ManufacturingProcess30 0.07039
## ManufacturingProcess28 0.06649
## ManufacturingProcess04 0.05921
## ManufacturingProcess34 0.05914
## BiologicalMaterial10   0.05645
## ManufacturingProcess25 0.05604
```

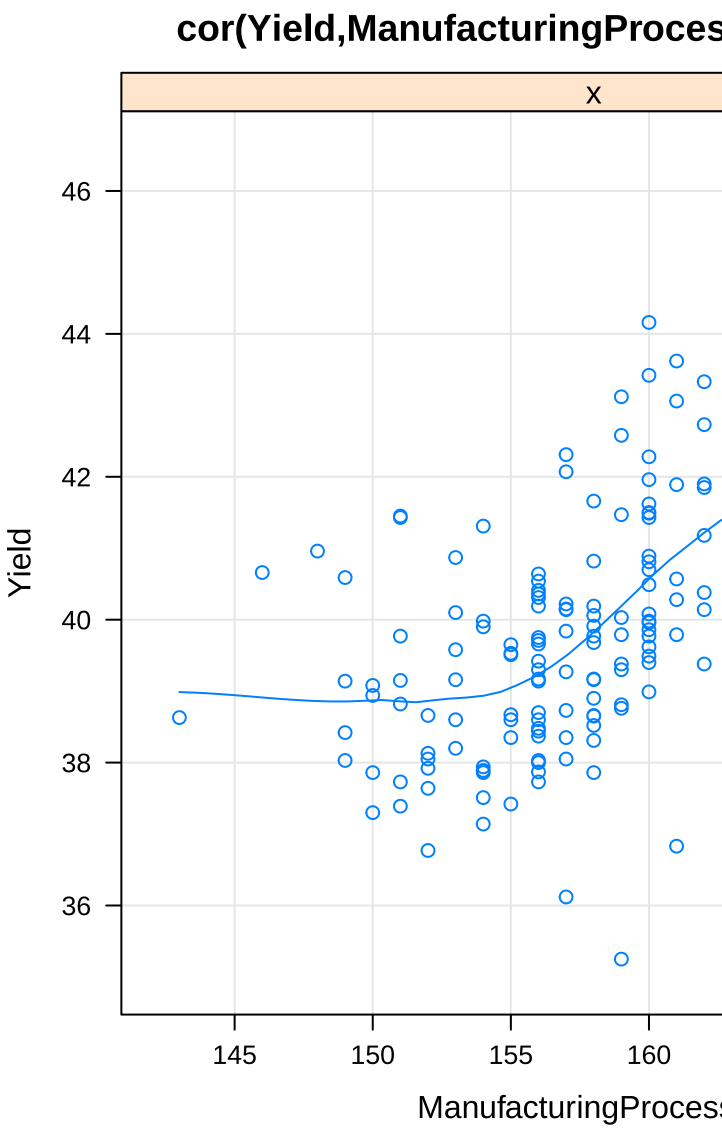
```r
plot(plsImp, top = 20,
     scales = list(y = list(cex = .75)),
     main="Importance of predictor variables for ChemicalManufacturingProcess")
```



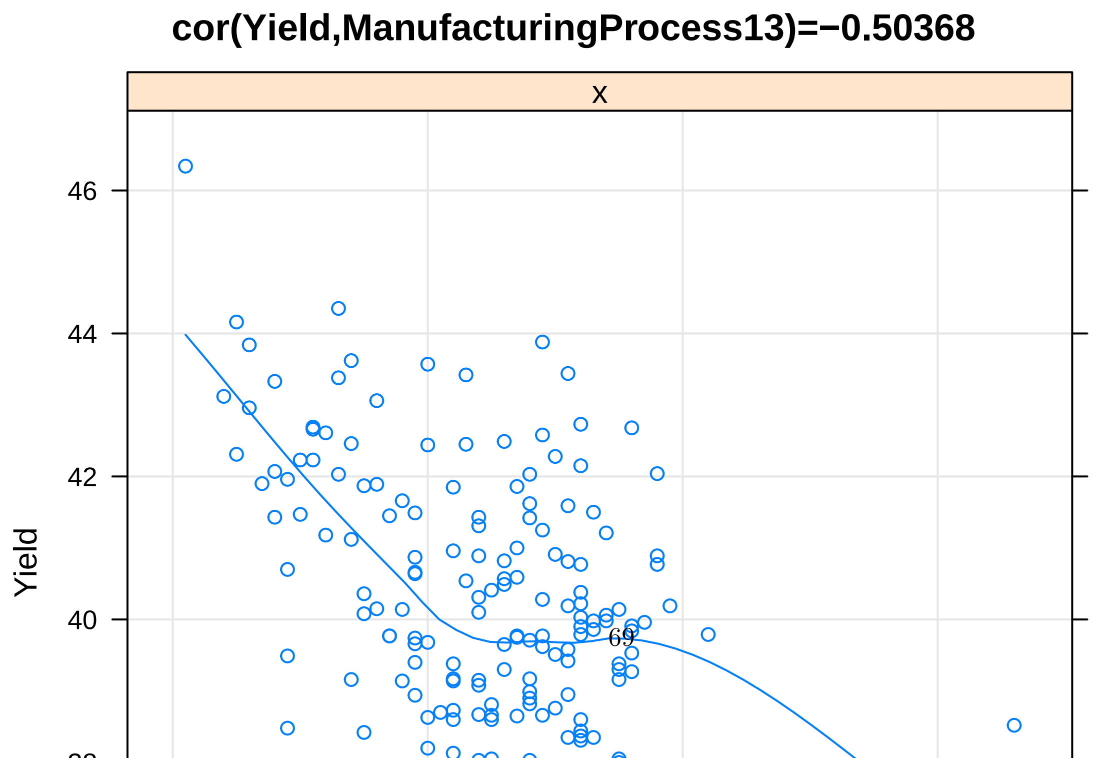
Do either the biological or process predictors dominate the list? The Manufacturing Process predictors dominate the list.

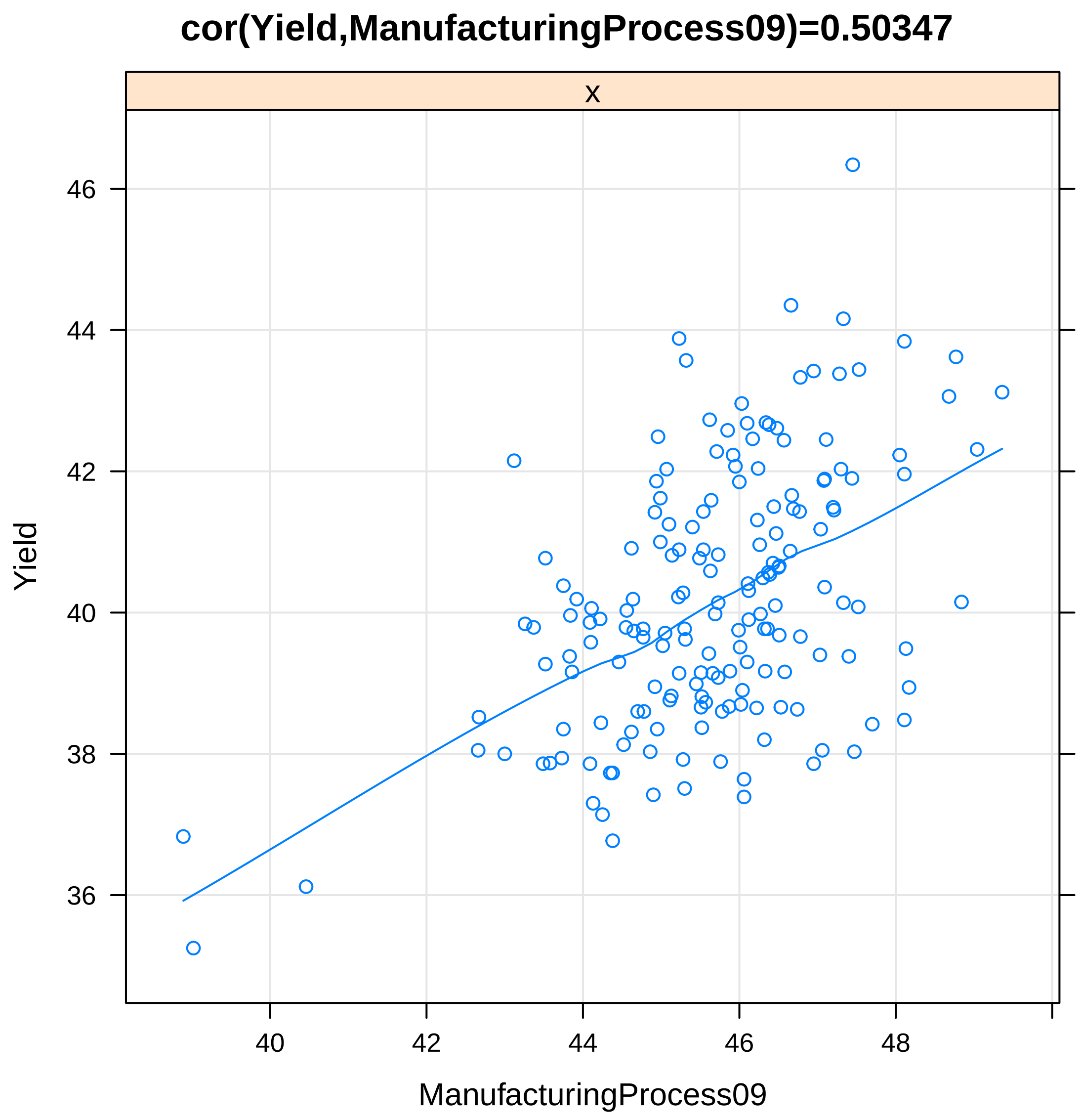
(f) Explore the relationships

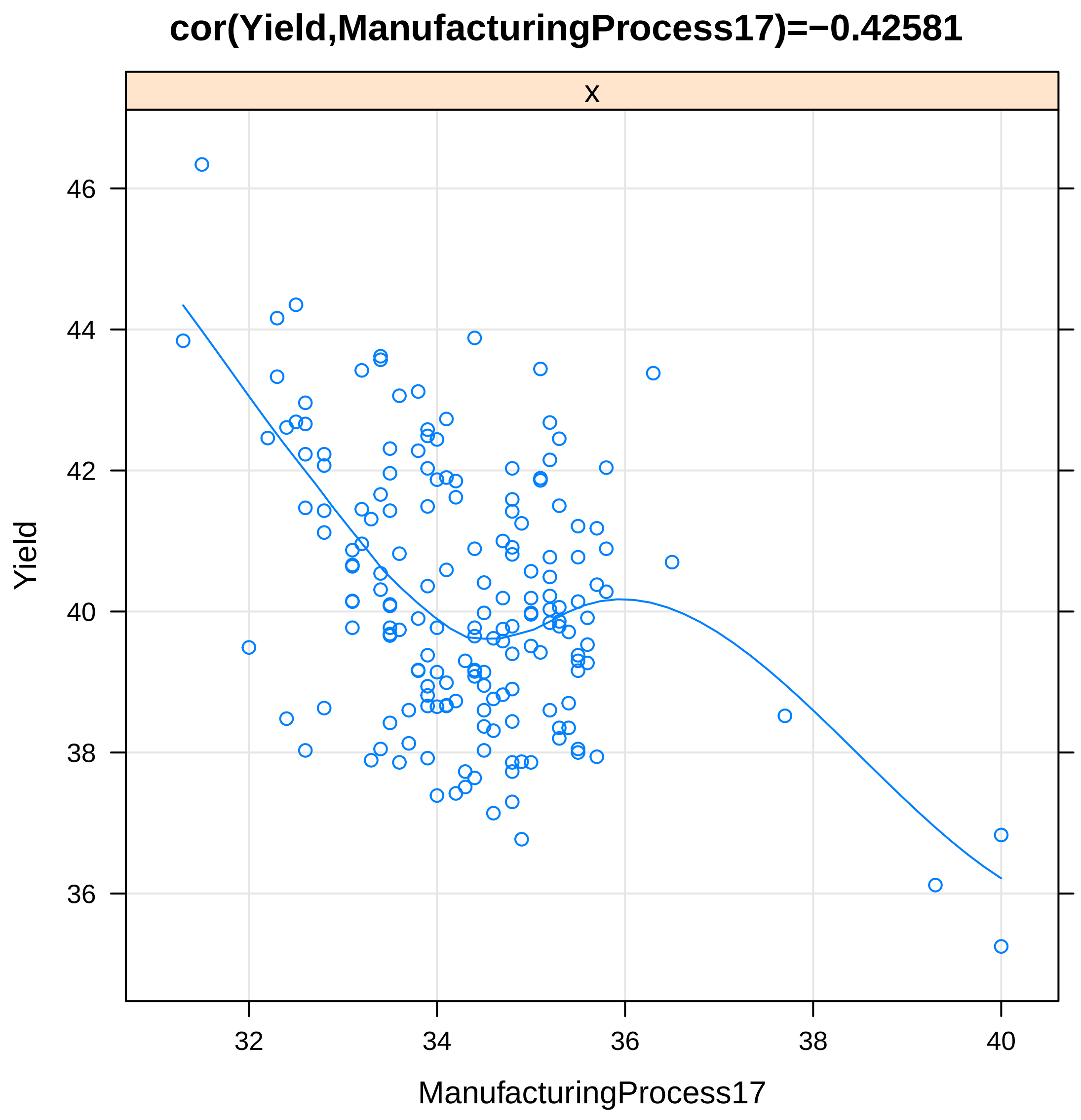
```r
#### Top ten predictors (by "importance")
topnames <- rownames(plsImp[["importance"]])[order(plsImp[["importance"]][["Overall"]],
                                                  decreasing = T)][1:10]

topcor=c()
#### Loop through top ten predictors
for (i in topnames) {
  #print(cor(yield,predictors[i]))
  topcor[i]=cor(yield,predictors[i])
  print(featurePlot(
    x = m_ChemicalManufacturingProcess[, i],
    y = m_ChemicalManufacturingProcess[, 1],
    between = list(x = 1, y = 1),
    type = c("g", "p", "smooth"),
    main=paste0("cor(Yield,",i,")=",round(topcor[i],5)),
    labels=c(i,"Yield")
  ))
}
```

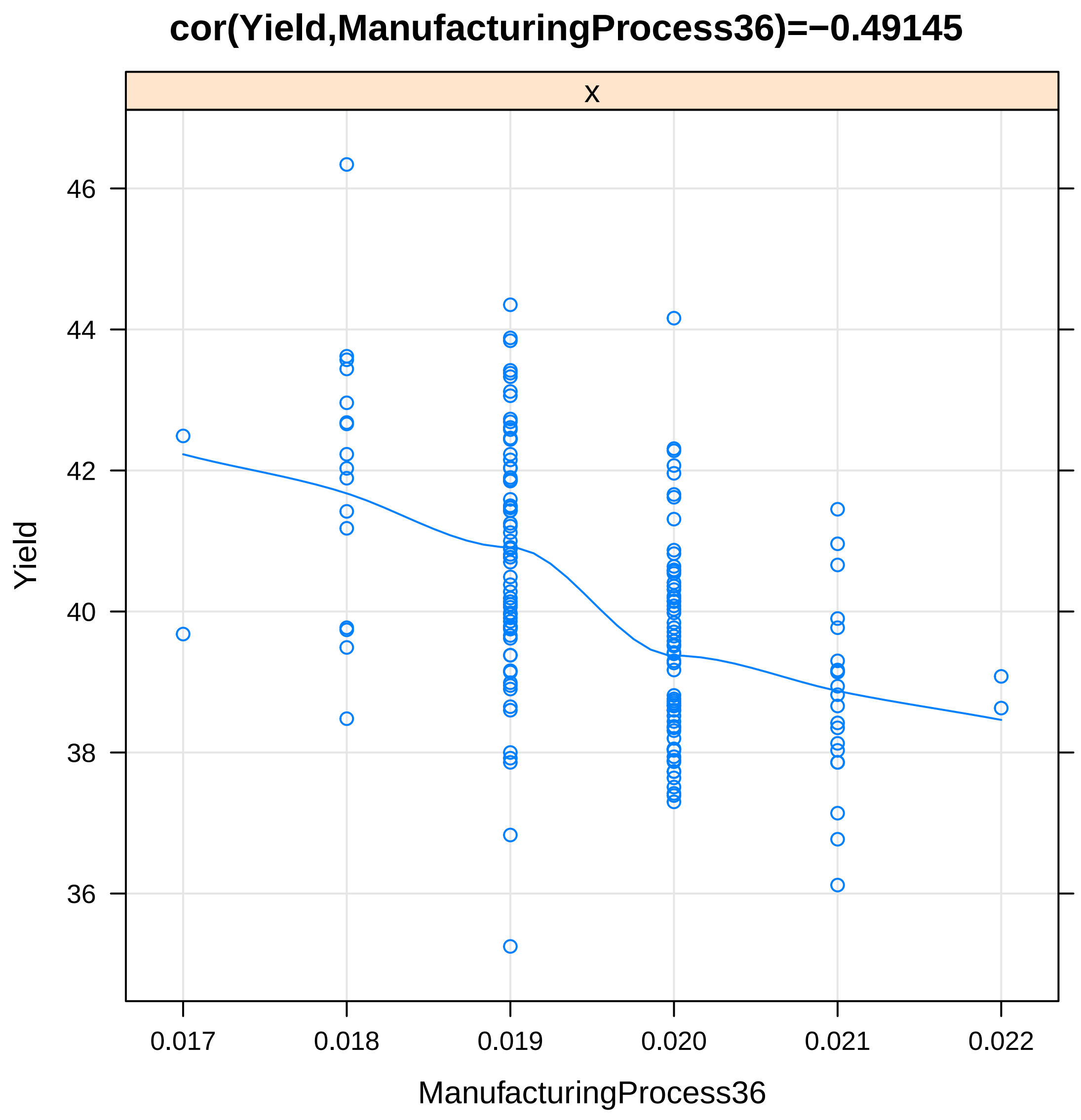


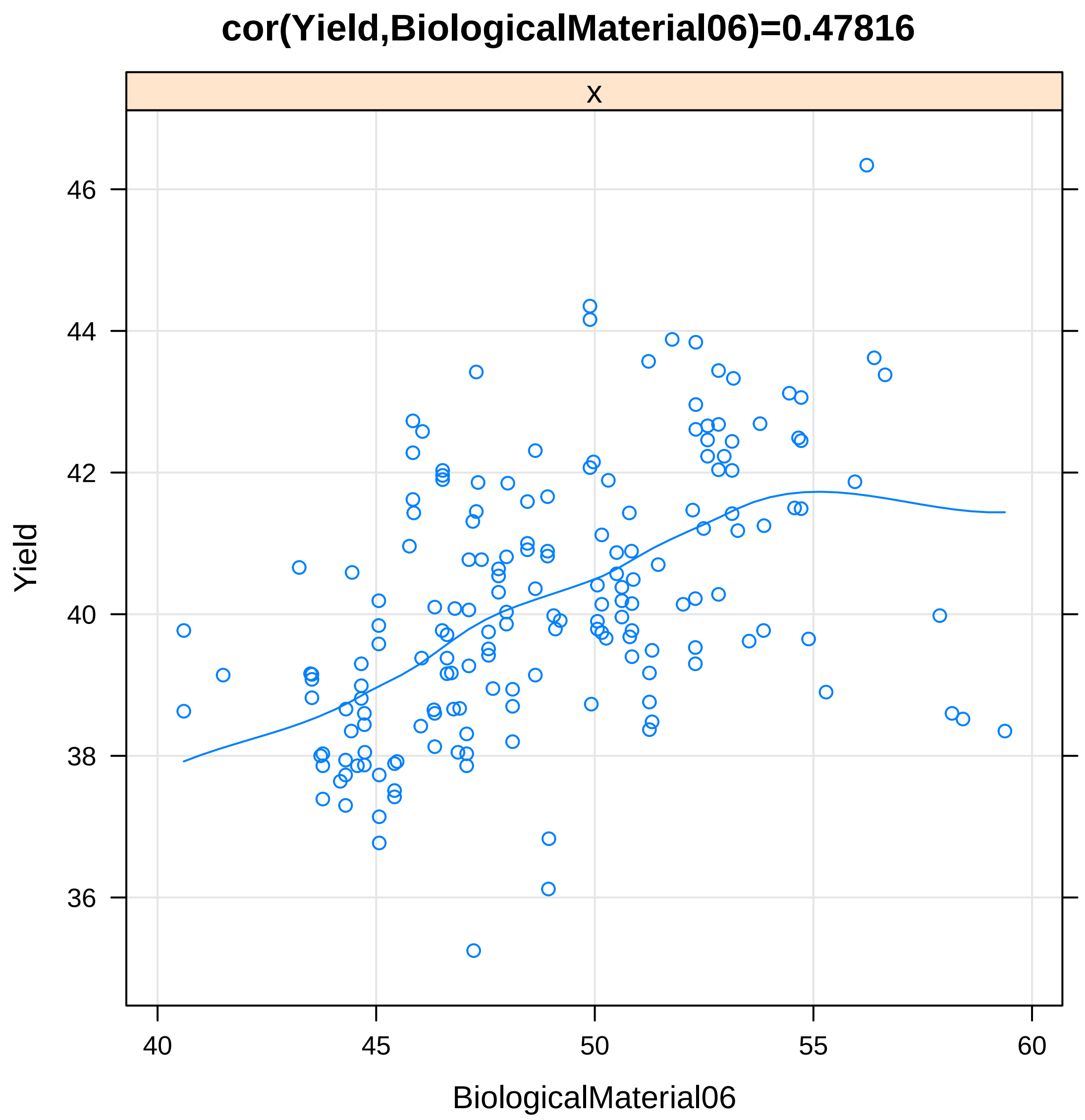
between each of the top predictors and the response.

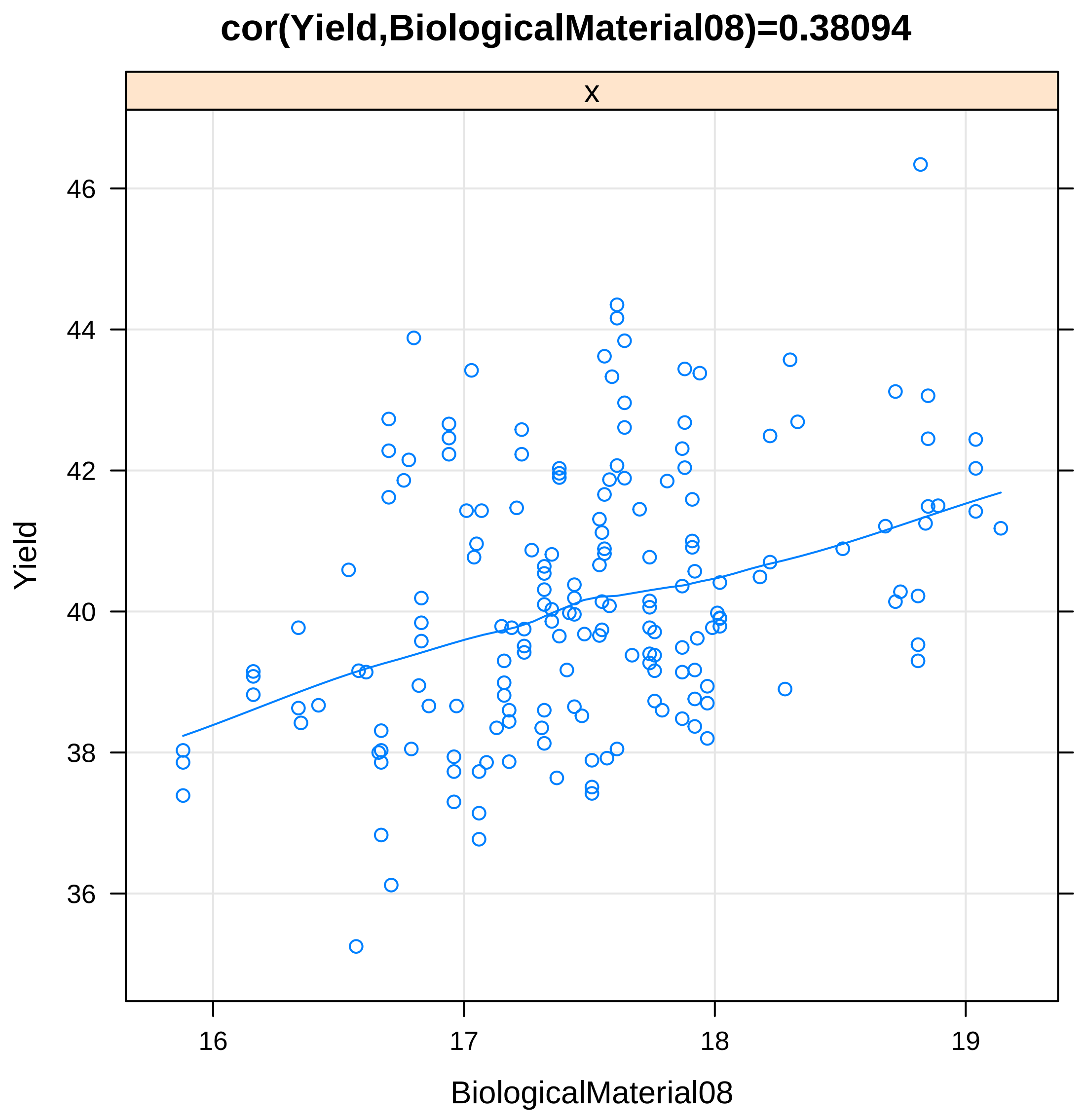



cor(Yield,ManufacturingProcess09)=0.50347
x
Yield
ManufacturingProcess09
36
38
40
42
44
46
40
42
44
46
48

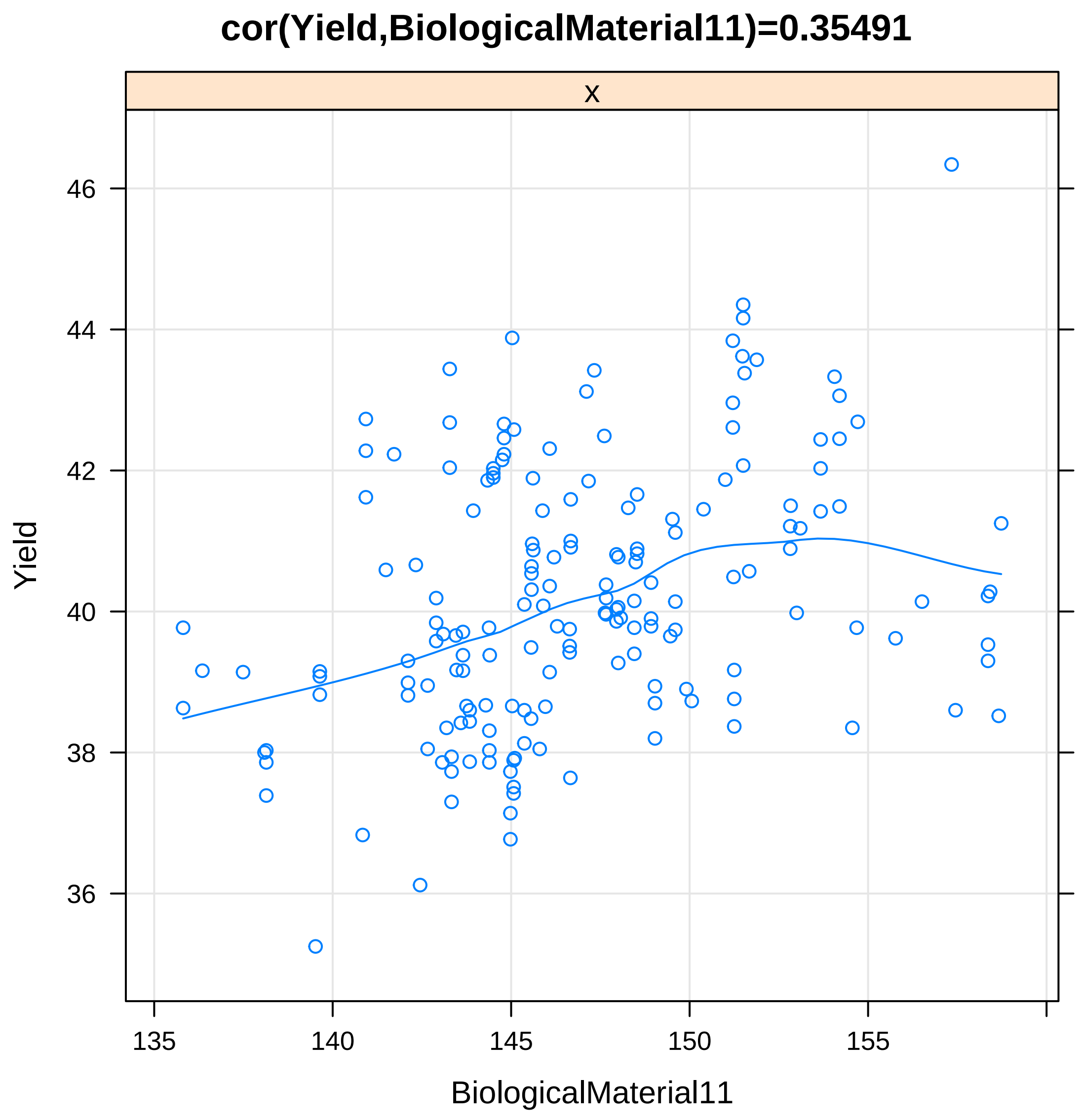

cor(Yield,ManufacturingProcess17)=−0.42581
x
Yield
46
44
42
40
38
36
32
34
36
38
40
ManufacturingProcess17

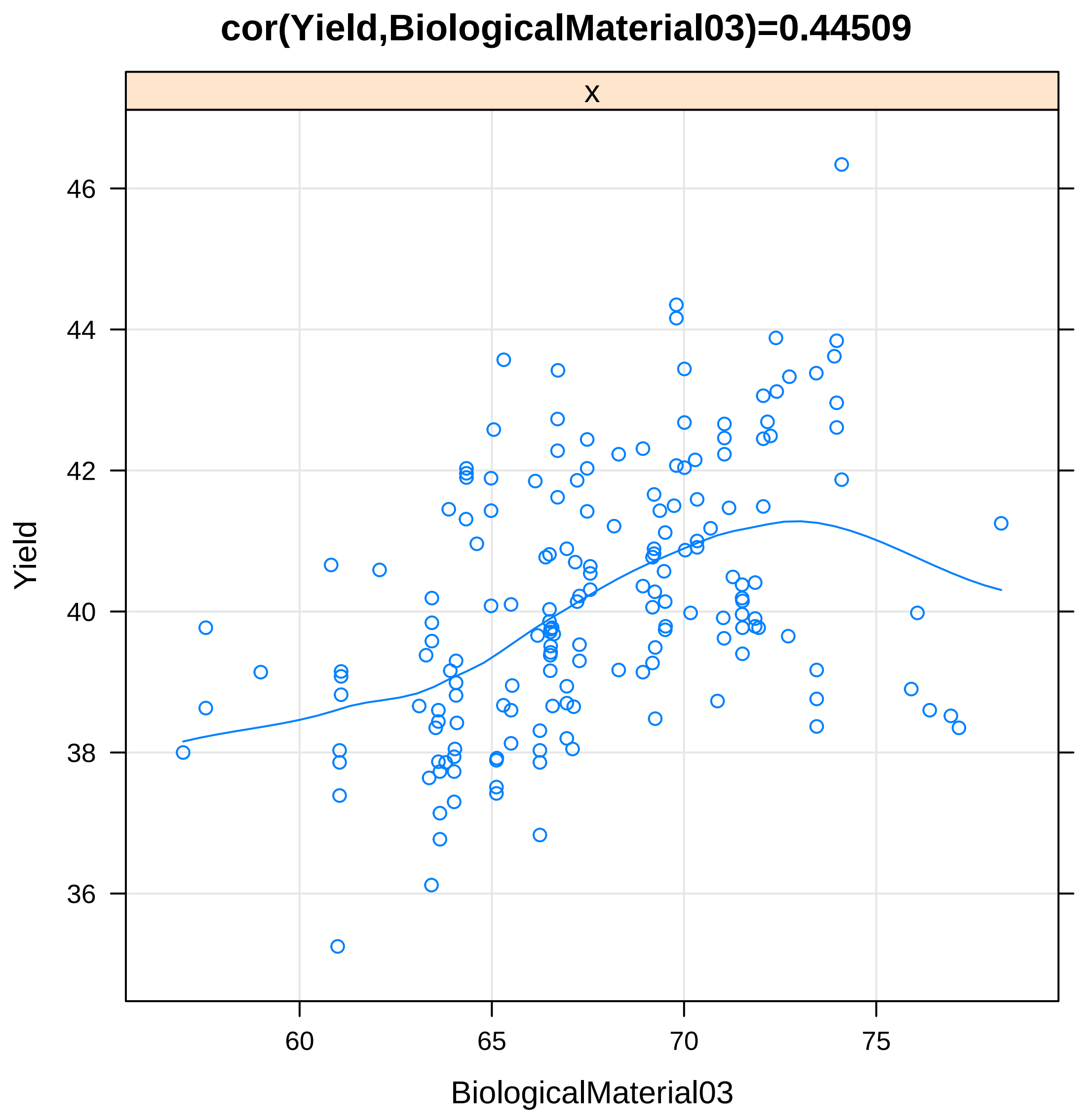

cor(Yield,ManufacturingProcess36)=−0.49145
x
Yield
46
44
42
40
38
36
0.017
0.018
0.019
0.020
0.021
0.022
ManufacturingProcess36

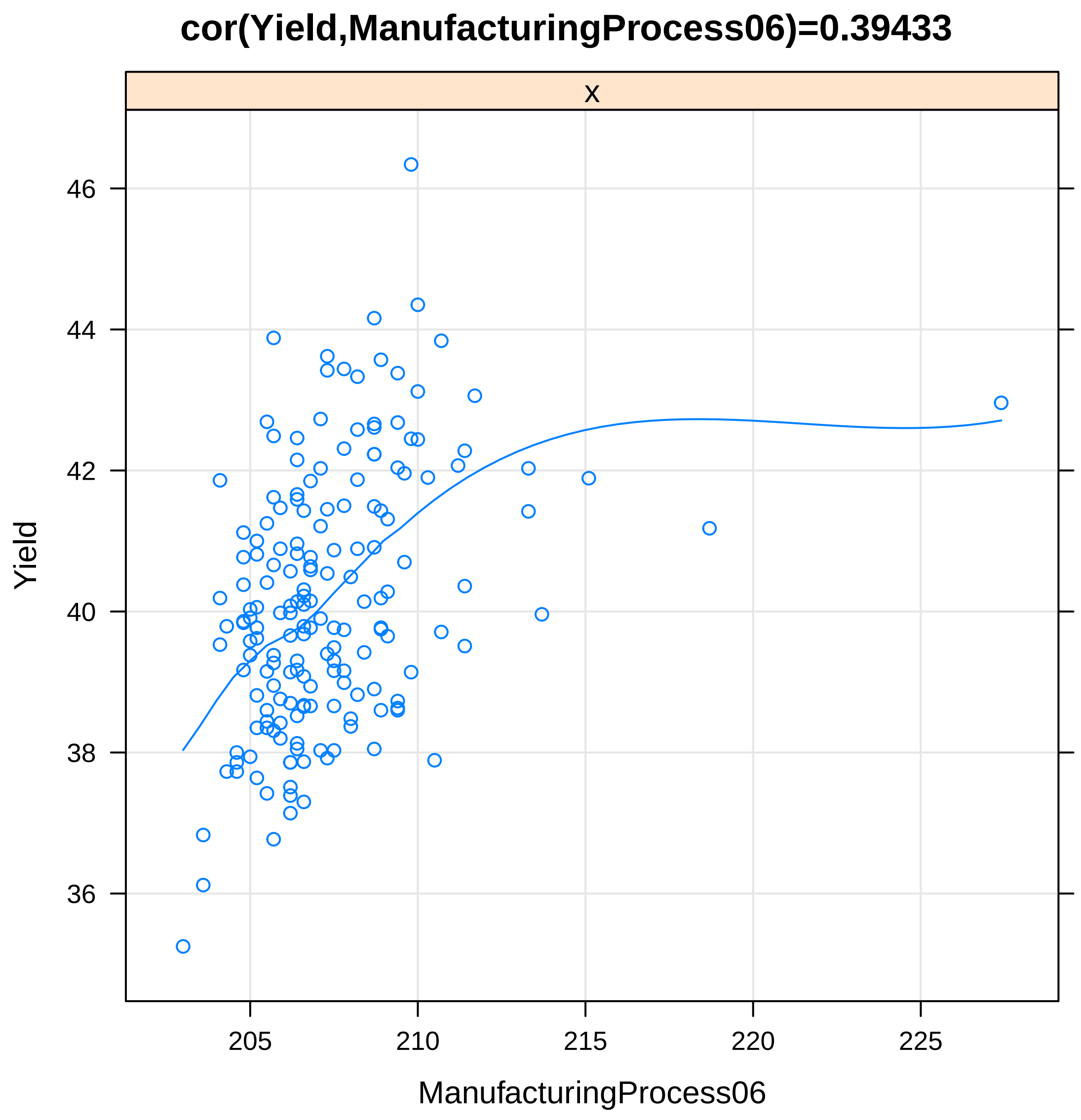

cor(Yield,BiologicalMaterial06)=0.47816
x
Yield
46
44
42
40
38
36
40
45
50
55
60
BiologicalMaterial06



cor(Yield,BiologicalMaterial08)=0.38094
x
Yield
BiologicalMaterial08
36
38
40
42
44
46
16
17
18
19


cor(Yield,BiologicalMaterial11)=0.35491
x
Yield
46
44
42
40
38
36
135
140
145
150
155
BiologicalMaterial11


cor(Yield,BiologicalMaterial03)=0.44509
x
Yield
46
44
42
40
38
36
60
65
70
75
BiologicalMaterial03



```
topcor <- as.matrix(topcor)
colnames(topcor) <- "Correlation with yield"


topcor %>%
  kable(caption = "Correlation between yield and most important predictors") %>%
  kable_styling(c("bordered","striped"),full_width = F)
```

Table 4: Correlation between yield and most important predictors

	Correlation with yield
ManufacturingProcess32	0.6083321
ManufacturingProcess13	-0.5036797
ManufacturingProcess09	0.5034705
ManufacturingProcess17	-0.4258069
ManufacturingProcess36	-0.4914450
BiologicalMaterial06	0.4781634
BiologicalMaterial08	0.3809402
BiologicalMaterial11	0.3549143
BiologicalMaterial03	0.4450860
ManufacturingProcess06	0.3943318

How could this information be helpful in improving yield in future runs of the manufacturing process? For the `ManufacturingProcess` predictors which display **high positive correlation** with `Yield`, such as 32, 09, and 06, it would be benefical to **increase** usage of such processes, as doing so should cause the yield to increase.

On the other hand, for those `ManufacturingProcess` predictors which display **negative correlation** with `Yield`, such as 13, 17, and 36, it would be beneficial, if possible, to curtail or otherwise **decrease** usage of such processes, as they cause the yield to decrease, so omitting or less reliance on such processes may cause an overall increase in the yield.