Data 624_Exercise_HW7

Jiaxin Zheng

2025-04-06

Kuhn and Johnson, Chapter 6

Exercise 6.2

- Developing a model to predict permeability (see Sect. 1.4) could save significant resources for a pharmaceutical company, while at the same time more rapidly identifying molecules that have a sufficient permeability to become a drug:

```
library(AppliedPredictiveModeling)
```

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

```
## Warning: package 'AppliedPredictiveModeling' was built under R version 4.4.3
```

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2

## Warning: package 'readr' was built under R version 4.4.3

## Warning: package 'dplyr' was built under R version 4.4.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.4.3
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.3
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
data(permeability)
```

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

(b) The fingerprint predictors indicate the presence or absence of substructures of a molecule and are often sparse meaning that relatively few of the molecules contain each substructure. Filter out the predictors that have low frequencies using the nearZeroVar function from the caret package. How many predictors are left for modeling?

- Answer: there are 388 predictors are left for modeling.

```
# Check for the structure of fingerprints
dim(fingerprints)
```

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

```
# filter out near-zero variance predictors
nzv <- nearZeroVar(fingerprints)

# keep predictors
filtered_fingerprints <- fingerprints[, -nzv]
ncol(filtered_fingerprints)
```

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

(c) Split the data into a training and a test set, pre-process the data, and tune a PLS model. How many latent variables are optimal and what is the corresponding resampled estimate of R^2?

- Answer: The final value used for the model was ncomp 9 and the R^2 is 0.5420556

```
set.seed(1234567)
# Selecting 80% of data as sample
train_index <- createDataPartition(permeability, p = 0.8, list=FALSE)
X_train <- filtered_fingerprints[train_index, ]
X_test  <- filtered_fingerprints[-train_index, ]

y_train <- permeability[train_index,]
y_test <- permeability[-train_index,]
```

```
set.seed(1234567)
# Set up training control with 10-fold cross-validation
ctrl <- trainControl(method = "cv", number = 10)

# Train a PLS model
pls_model <- train(
  x = X_train,
  y = y_train,
  method = "pls",
  preProcess = c("center", "scale"),
  tuneLength = 20,
  trControl = ctrl
)

print(pls_model)
```

```
## Partial Least Squares
##
## 133 samples
## 388 predictors
##
## Pre-processing: centered (388), scaled (388)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 121, 121, 119, 120, 121, 119, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared   MAE
##    1     13.01013  0.3099773  9.557409
##    2     11.54055  0.4741625  8.108504
##    3     11.60880  0.4808389  8.646855
##    4     11.69601  0.4747452  8.745717
##    5     11.54479  0.4965186  8.909402
##    6     11.28312  0.5189130  8.597759
##    7     11.18076  0.5333536  8.500229
##    8     11.11353  0.5266841  8.549144
##    9     11.01671  0.5420556  8.529739
##   10     11.16218  0.5362516  8.531159
##   11     11.55680  0.5145957  8.707311
```

```
##   12     11.63002  0.5109617  8.837075
##   13     11.80589  0.5057671  9.095128
##   14     12.00103  0.4988392  9.269210
##   15     12.17042  0.4905478  9.384682
##   16     12.50724  0.4746625  9.563831
##   17     12.67378  0.4654585  9.682942
##   18     12.78672  0.4594005  9.651136
##   19     12.84184  0.4599673  9.631870
##   20     12.79707  0.4664975  9.539671
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 9.
```

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

- Answer: The test set estimate R^2 is 0.5326435

```
# Predict permeability on X test
predictions <- predict(pls_model, X_test)

test_results <- postResample(predictions, y_test)

test_results
```

```
##       RMSE   Rsquared        MAE
## 11.4987267  0.5326435  8.2999059
```

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

- I tested Ridge model & Elastic Net.
- R^2 for Ride model is 0.5819108, and RMSE is 10.3519819.
- Elastic Net's R^2 is 0.624194, RMSE is 9.2244272.

- PLS's R^2 is 0.5326435, RMSE is 11.4987267
- I think Elastic Net model is better predictive performance.

```
set.seed(100)
ridge_model <- train(
  x = X_train,
  y = y_train,
  method = "ridge",
  preProcess = c("center", "scale"),
  tuneLength = 20,
  trControl = ctrl
)
```

Ridge Model

```
## Warning: model fit failed for Fold09: lambda=0.0000000 Error in if (zmin < gamhat) { : missing value

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

```
print (ridge_model)
```

```
## Ridge Regression
##
## 133 samples
## 388 predictors
##
## Pre-processing: centered (388), scaled (388)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 120, 120, 119, 121, 120, 119, ...
## Resampling results across tuning parameters:
##
##   lambda        RMSE          Rsquared   MAE
##   0.0000000000  1.466186e+01  0.3096589  1.036603e+01
##   0.0001000000  1.489264e+05  0.1482612  6.224874e+04
##   0.0001467799  1.842330e+04  0.1672330  1.037027e+04
##   0.0002154435  4.655252e+03  0.2024351  2.217720e+03
##   0.0003162278  1.052224e+06  0.2204926  5.869168e+05
##   0.0004641589  3.630209e+04  0.1793461  1.825388e+04
##   0.0006812921  1.441247e+04  0.1549150  8.266751e+03
##   0.0010000000  1.390319e+05  0.1916739  9.020470e+04
##   0.0014677993  9.870285e+02  0.2789141  5.936280e+02
##   0.0021544347  5.273378e+03  0.2079357  3.485262e+03
##   0.0031622777  2.653689e+03  0.2033933  1.945875e+03
##   0.0046415888  6.299458e+02  0.2989492  3.882157e+02
##   0.0068129207  2.212747e+04  0.3247655  1.275142e+04
##   0.0100000000  1.478391e+01  0.3490833  1.079271e+01
##   0.0146779927  1.401175e+01  0.3658026  1.017683e+01
##   0.0215443469  1.343015e+01  0.3835055  9.745720e+00
##   0.0316227766  1.295761e+01  0.4005511  9.419151e+00
##   0.0464158883  1.256128e+01  0.4178368  9.178205e+00
##   0.0681292069  1.228791e+01  0.4335939  9.037986e+00
##   0.1000000000  1.203816e+01  0.4524199  8.953531e+00
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was lambda = 0.1.
```

```
ridge_pred <- predict(ridge_model, X_test)
ridge_r2 <- postResample(ridge_pred, y_test)

ridge_r2
```

```
##       RMSE   Rsquared        MAE
## 11.1596570  0.5605117  8.1195016
```

```
enetGrid <- expand.grid(.lambda = c(0, 0.01, .1),
                        .fraction = seq(0.05, 1, length =20))
set.seed(100)

enetTune <- train(
  x = X_train,
  y = y_train,
  method = "enet",
  preProcess = c("center", "scale"),
  tuneGrid = enetGrid,
  trControl = ctrl
)
```

Elastic Net

```
## Warning: model fit failed for Fold09: lambda=0.00, fraction=1 Error in if (zmin < gamhat) { : missing
```

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

```
print(enetTune)
```

```
## Elasticnet
##
## 133 samples
## 388 predictors
##
## Pre-processing: centered (388), scaled (388)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 120, 120, 119, 121, 120, 119, ...
## Resampling results across tuning parameters:
##
##   lambda  fraction  RMSE      Rsquared   MAE
##   0.00    0.05      12.21029  0.4969496   9.165548
##   0.00    0.10      11.85653  0.4819697   8.348031
##   0.00    0.15      11.72907  0.4815379   8.254107
##   0.00    0.20      11.81018  0.4719727   8.479348
##   0.00    0.25      11.79360  0.4694946   8.628634
##   0.00    0.30      11.87748  0.4582529   8.719240
##   0.00    0.35      11.95128  0.4482125   8.780863
##   0.00    0.40      12.09830  0.4349503   8.837634
##   0.00    0.45      12.25758  0.4208637   8.863680
##   0.00    0.50      12.29972  0.4176961   8.862903
##   0.00    0.55      12.37293  0.4128954   8.901096
##   0.00    0.60      12.47448  0.4078234   8.956424
##   0.00    0.65      12.65975  0.3971524   9.053050
##   0.00    0.70      12.89627  0.3840012   9.199290
##   0.00    0.75      13.15320  0.3706860   9.374415
##   0.00    0.80      13.44641  0.3557248   9.575425
##   0.00    0.85      13.72585  0.3418261   9.760259
##   0.00    0.90      14.03366  0.3285374   9.962159
```

```
##   0.00    0.95      14.32630  0.3193650  10.164481
##   0.00    1.00      14.66186  0.3096589  10.366031
##   0.01    0.05      12.05082  0.4673108   8.636344
##   0.01    0.10      11.75077  0.4707031   8.334112
##   0.01    0.15      11.81767  0.4512327   8.591487
##   0.01    0.20      11.72893  0.4501304   8.562705
##   0.01    0.25      11.59228  0.4600893   8.408376
##   0.01    0.30      11.69372  0.4522034   8.458347
##   0.01    0.35      11.93294  0.4320771   8.557577
##   0.01    0.40      12.26425  0.4103299   8.715820
##   0.01    0.45      12.56015  0.3932520   8.902246
##   0.01    0.50      12.80115  0.3825738   9.095135
##   0.01    0.55      12.97751  0.3775984   9.220209
##   0.01    0.60      13.15421  0.3755579   9.313258
##   0.01    0.65      13.32585  0.3748989   9.455510
##   0.01    0.70      13.50840  0.3725313   9.598040
##   0.01    0.75      13.71426  0.3696005   9.772057
##   0.01    0.80      13.97289  0.3639354  10.020574
##   0.01    0.85      14.19890  0.3596286  10.219863
##   0.01    0.90      14.40915  0.3561855  10.436951
##   0.01    0.95      14.60379  0.3527506  10.631341
##   0.01    1.00      14.78391  0.3490833  10.792710
##   0.10    0.05      12.13663  0.4625157   9.086294
##   0.10    0.10      11.87931  0.4749875   8.365144
##   0.10    0.15      11.73068  0.4827215   8.290836
##   0.10    0.20      11.68473  0.4759945   8.396060
##   0.10    0.25      11.66921  0.4715616   8.507026
##   0.10    0.30      11.62925  0.4711791   8.540243
##   0.10    0.35      11.50973  0.4763679   8.467290
##   0.10    0.40      11.42544  0.4809876   8.375470
##   0.10    0.45      11.39332  0.4835217   8.348163
##   0.10    0.50      11.42938  0.4814608   8.378857
##   0.10    0.55      11.50528  0.4763291   8.429835
##   0.10    0.60      11.58491  0.4712630   8.479586
##   0.10    0.65      11.65646  0.4678860   8.554847
##   0.10    0.70      11.71622  0.4655838   8.620075
##   0.10    0.75      11.77843  0.4631365   8.673421
##   0.10    0.80      11.83911  0.4607466   8.735685
##   0.10    0.85      11.89536  0.4584762   8.797261
##   0.10    0.90      11.94939  0.4564078   8.864538
##   0.10    0.95      11.99539  0.4543778   8.911208
##   0.10    1.00      12.03816  0.4524199   8.953531
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were fraction = 0.45 and lambda = 0.1.
```

```
enet_pred <- predict(enetTune, X_test)
enet_r2 <- postResample(enet_pred, y_test)

enet_r2
```

```
##      RMSE  Rsquared       MAE
## 10.475628  0.588572  7.594841
```

Exercise 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:

```
data("ChemicalManufacturingProcess")
```

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

```
sum(is.na(ChemicalManufacturingProcess))
```

(b) A small percentage of cells in the predictor set contain missing values. Use an imputation function to fill in these missing values (e.g., see Sect. 3.8).

```
## [1] 106
```

```
# Separate predictors and response
predictors <- ChemicalManufacturingProcess[, -ncol(ChemicalManufacturingProcess)]
response <- ChemicalManufacturingProcess$Yield

# Impute missing values using knn imputation
chem_pre <- preProcess(predictors, method = "knnImpute")
chem_imputed_data <- predict(chem_pre, predictors)

sum(is.na(chem_imputed_data))
```

```
## [1] 0
```

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

- The final value used for the model was ncomp = 3. R^2 is 0.11102077

```
set.seed(1234567)

Chem_train_index <- createDataPartition(response, p = 0.8, list = FALSE)

X_train <- chem_imputed_data[Chem_train_index, ]
X_test  <- chem_imputed_data[-Chem_train_index, ]
```

```r
y_train <- response[Chem_train_index]
y_test  <- response[-Chem_train_index]

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

```r
set.seed(1234567)
pls_model2 <- train(
  x = X_train,
  y = y_train,
  method = "pls",
  preProcess = c("center", "scale"),
  tuneLength = 20,
  trControl = ctrl
)

print(pls_model2)
```

```
## Partial Least Squares
##
## 144 samples
##  57 predictor
##
## Pre-processing: centered (57), scaled (57)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 129, 130, 130, 128, 130, 130, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared   MAE
##    1     1.33762547  0.5147399  1.05590762
##    2     1.27337738  0.6656675  0.89036831
##    3     0.82669807  0.8029996  0.66146713
##    4     0.64549384  0.8916696  0.53164814
##    5     0.45855923  0.9387938  0.37622221
##    6     0.36459882  0.9607000  0.30499229
##    7     0.29853744  0.9708600  0.22914447
##    8     0.25938120  0.9692650  0.16740124
##    9     0.20498293  0.9825064  0.13604107
##   10     0.16639598  0.9889704  0.10946761
##   11     0.10635297  0.9968002  0.08158253
##   12     0.08696361  0.9976567  0.06181070
##   13     0.09099919  0.9961488  0.05668793
##   14     0.08551475  0.9958948  0.04977238
##   15     0.06335773  0.9980943  0.03950324
##   16     0.06206046  0.9978372  0.03710197
##   17     0.06142948  0.9975157  0.03456781
##   18     0.06954597  0.9957570  0.03495620
##   19     0.07172404  0.9946032  0.03279033
##   20     0.05903490  0.9963391  0.02722846
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 20.
```

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

- RMSE is 0.11410910, and R^2 is 0.9961308

```
# Predict on test set
predictions2 <- predict(pls_model2, X_test)

test_results2 <- postResample(predictions2, y_test)

test_results2
```

```
##       RMSE   Rsquared        MAE
## 0.11410910 0.99613080 0.04373019
```

```
list <- varImp(pls_model2)
```

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

```
## Warning: package 'pls' was built under R version 4.4.3
```

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

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

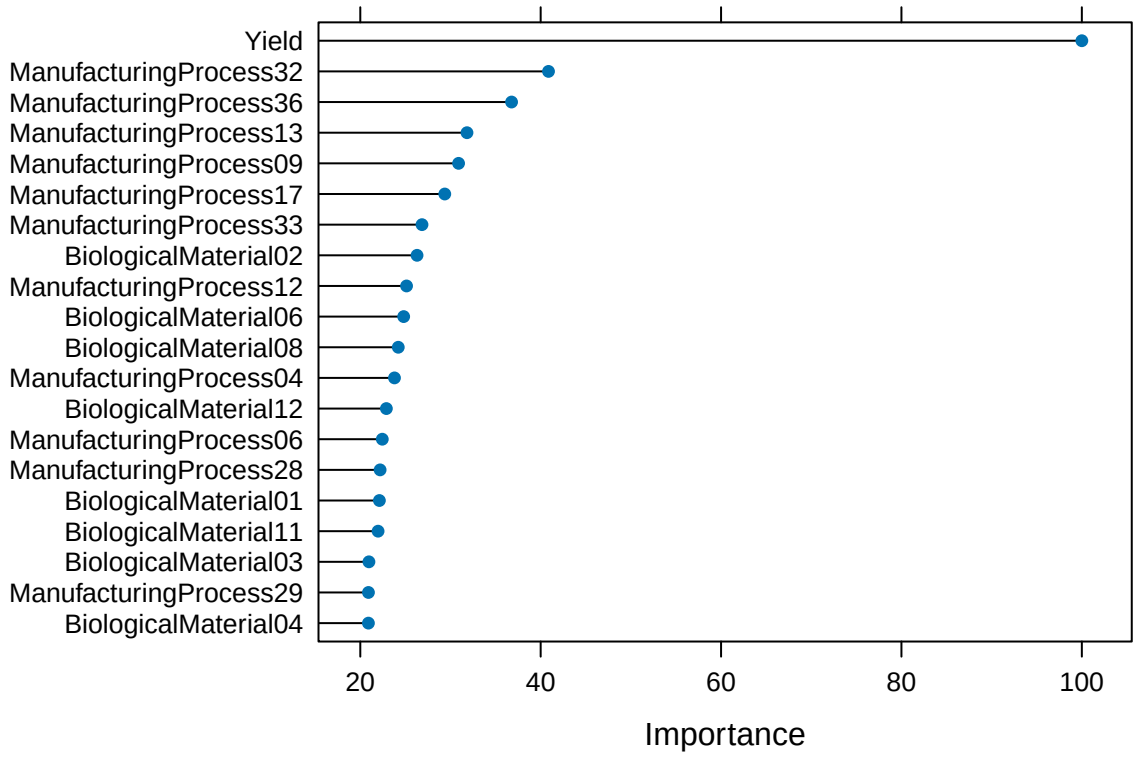
```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
print(list)
```

```
## pls variable importance
##
##   only 20 most important variables shown (out of 57)
##
##                          Overall
## Yield                     100.00
## ManufacturingProcess32     40.87
## ManufacturingProcess36     36.77
## ManufacturingProcess13     31.83
## ManufacturingProcess09     30.90
## ManufacturingProcess17     29.37
## ManufacturingProcess33     26.84
## BiologicalMaterial02       26.29
## ManufacturingProcess12     25.13
```

```
## BiologicalMaterial06     24.81
## BiologicalMaterial08     24.21
## ManufacturingProcess04   23.79
## BiologicalMaterial12     22.89
## ManufacturingProcess06   22.44
## ManufacturingProcess28   22.20
## BiologicalMaterial01     22.12
## BiologicalMaterial11     21.97
## BiologicalMaterial03     20.96
## ManufacturingProcess29   20.91
## BiologicalMaterial04     20.90
```

```
plot(list, top = 20)
```



(f) Explore the relationships between each of the top predictors and the response. How could this information be helpful in improving yield in future runs of the manufacturing process? - It shows BiologicalMaterial01 & 06 have the strongest correlation among the manufacturing predictors at 0.95, and follow by ManufacturingProcess32 & 33 at 0.86

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.4.2
```

```
## corrplot 0.95 loaded
```

```
##
## Attaching package: 'corrplot'
```

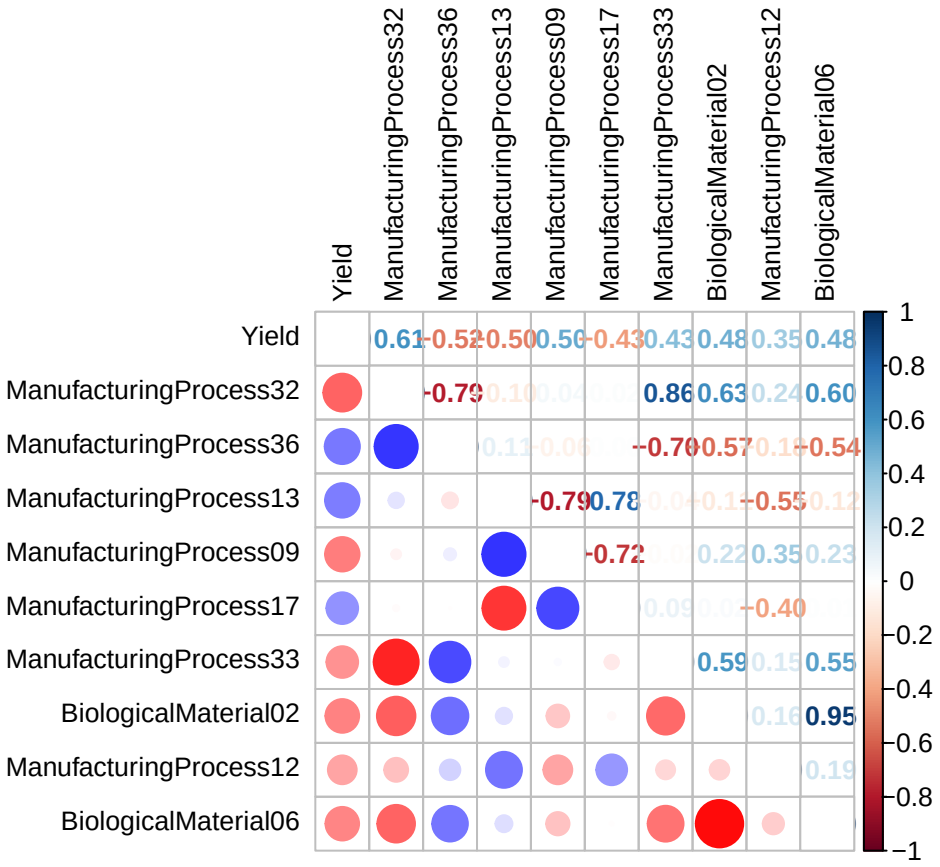
```
## The following object is masked from 'package:pls':
##
##     corrplot
```

```r
chem_data <- data.frame(chem_imputed_data, Yield = response)

# Select the top predictors along with Yield.
top_corr_vars <- chem_data[, c(
  "Yield",
  "ManufacturingProcess32", "ManufacturingProcess36", "ManufacturingProcess13",
  "ManufacturingProcess09", "ManufacturingProcess17", "ManufacturingProcess33",
  "BiologicalMaterial02", "ManufacturingProcess12", "BiologicalMaterial06"
)]

# Compute the correlation matrix using pairwise complete observations.
corr_matrix <- cor(top_corr_vars, use = "pairwise.complete.obs")


corrplot.mixed(
  corr_matrix,
  upper = "number",
  lower = "circle",
  tl.pos = "lt",
  tl.col = "black",
  number.cex = 0.8,
  tl.cex = 0.8,
  lower.col = colorRampPalette(c("blue", "white", "red"))(200)
)
```


Yield
ManufacturingProcess32
ManufacturingProcess36
ManufacturingProcess13
ManufacturingProcess09
ManufacturingProcess17
ManufacturingProcess33
BiologicalMaterial02
ManufacturingProcess12
BiologicalMaterial06
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1