Week3, Assignment 1

Gabi Rivera

```
library(caret)
library(AppliedPredictiveModeling)
library(dplyr)
library(ggplot2)
library(factoextra)
library(corrplot)
library(car)
library(knitr)
library(gt)
library(naniar)
library(mice)
```

Problem 3.1 (30 points)

A Tecator Infratec Food and Feed Analyzer instrument was used to analyze 215 samples of meat across 100 frequencies. In addition to an IR profile, analytical chemistry determined the percent content of water, fat, and protein for each sample. If we can establish a predictive relationship between IR spectrum and fat content, then food scientists could predict a sample's fat content with IR instead of using analytical chemistry. This would provide costs savings since analytical chemistry is a more expensive, time-consuming process

3.1.a Load the data

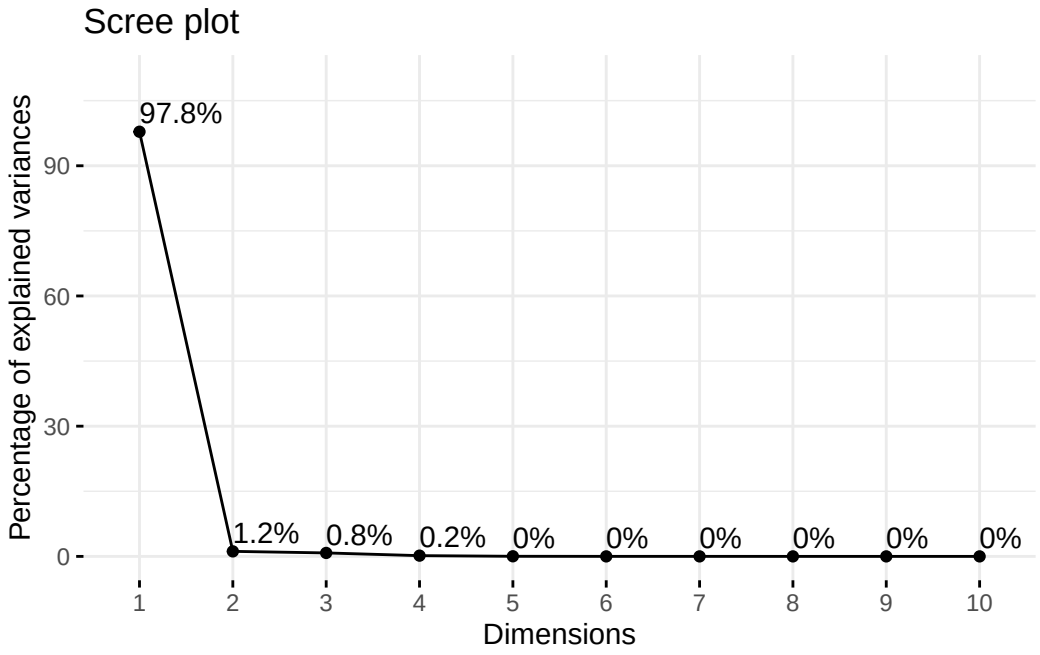
```
# Upload data tecator data frames
data(tecator)
?tecator
# Combine IR predictors and fat content outcome into a new data frame
fat_outcome <- endpoints[,2]
IR_fat <- cbind(absorp, Fat = fat_outcome)
IR_fat <- as.data.frame(IR_fat)
```

The matrix absorp contains the 100 absorbance values for the 215 samples, while matrix endpoints contain the percent of moisture, **fat**, and protein in columns 1–3, respectively.

3.1.b (5 points)

Use PCA to determine the effective dimension of these data. What is the effective dimension(it's the number of principal components needed to explain the majority of the variance in the data)?

```
# Perform PCA on the absorp data frame
pca_result <- prcomp(IR_fat, scale = TRUE)
#summary(pca_result)

# Visualize variance explained by each principal component
fviz_eig(pca_result, addlabels = TRUE, ylim = c(0, 110), geom = "line")
```




```
# Determine effective dimension
var_explained <- pca_result$sdev^2 / sum(pca_result$sdev^2)
cumsum_var_explained <- cumsum(var_explained)
effective_dim <- which.max(cumsum_var_explained >= 0.9)

cat("The effective dimension (number of principal components needed to
    explain 90% of variance) is:", effective_dim)
```

```
The effective dimension (number of principal components needed to
    explain 90% of variance) is: 1
```

3.1.c (20 points)

Split the data into a training and a validation set using a resampling technique, pre-process the data by centering and scaling, and build linear regression, partial least squares, ridge regression, lasso regression and elastic net models described in this chapter. For those models with tuning parameters, what are the optimal values of the tuning parameter(s)?

```r
# Remove highly correlated predictors
correlation_matrix <- cor(IR_fat[, -1])
highly_correlated <- findCorrelation(correlation_matrix, cutoff = 0.9998)
highly_correlated_names <- colnames(IR_fat[, -1])[highly_correlated]
IR_fat_reduced <- IR_fat[, -highly_correlated]
```

```r
# Create a list of N folds K-Fold CV
set.seed(123)
ctrl <- trainControl(method = "cv", number = 3)
folds <- createFolds(IR_fat_reduced$Fat, k = 3)

models <- list()
for(i in 1:3) {
  train_data <- IR_fat_reduced[-folds[[i]], ]
  test_data <- IR_fat_reduced[folds[[i]], ]

# Build Linear Regression Model
lm_model <- train(Fat ~ .,
                  data = train_data,
                  method = "lm",
                  trControl = ctrl,
                  preProcess = c("center", "scale"))
models[[paste("lm_Model", i, sep = "_")]] <- lm_model

# Create Partial Least Squeare Model
pls_model <- train(Fat ~ .,
                   data = train_data,
                   method = "pls",
                   trControl = ctrl,
                   preProcess = c("center", "scale"))
models[[paste("pls_Model", i, sep = "_")]] <- pls_model
```

```r
# Create Ridge Regression Model
ridgeGrid <- expand.grid(lambda = seq(0, 0.1, length = 5))
ridge_model <- train(Fat ~ .,
                     data = train_data,
                     method = "ridge",
                     tuneGrid = ridgeGrid,
                     trControl = ctrl,
                     preProcess = c("center", "scale"))
models[[paste("ridge_Model", i, sep = "_")]] <- ridge_model

# Create Lasso Regression Model
lasGrid <- expand.grid(fraction = seq(0.001, 1, length = 5))
lasso_model <- train(Fat ~ .,
                     data = train_data,
                     method = "lasso",
                     tuneGrid = lasGrid,
                     trControl = ctrl,
                     preProcess = c("center", "scale"))
models[[paste("lasso_model", i, sep = "_")]] <- lasso_model

#Create Elastic Net Models
enetGrid <- expand.grid(alpha = seq(0, 1, by = 0.4),
                        lambda = seq(0, 1, by = 0.4))
elastic_net_model <- train(Fat ~ .,            # Outcome variable and all predictors
                           data = train_data,      # Dataset
                           method = "glmnet",     # Elastic Net method
                           trControl = ctrl,      # Cross-validation control
                           tuneGrid = enetGrid, # Grid of tuning parameters (alpha)
                           preProcess = c("center", "scale"))  # Standardize predictors
models[[paste("elastic_net_model", i, sep = "_")]] <- elastic_net_model}
models
```

```
$lm_Model_1
Linear Regression

144 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
```

```
Resampling results:

  RMSE      Rsquared   MAE
  4.604237  0.8822562  3.618991

Tuning parameter 'intercept' was held constant at a value of TRUE

$pls_Model_1
Partial Least Squares

144 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared   MAE
  1      11.764918  0.1403162  9.693155
  2       7.479541  0.6410243  5.565003
  3       5.142096  0.8217445  3.996089

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 3.

$ridge_Model_1
Ridge Regression

144 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  lambda  RMSE      Rsquared   MAE
  0.000   4.510401  0.8783840  3.586540
  0.025   7.652608  0.7161582  6.170189
  0.050   8.890422  0.5995846  7.195053
  0.075   9.527305  0.5171996  7.731461
  0.100   9.921636  0.4574669  8.074031
```

```
RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.

$lasso_model_1
The lasso

144 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  fraction  RMSE       Rsquared   MAE
  0.00100   11.901749  0.1920902  10.094779
  0.25075    4.385660  0.8776944   3.578048
  0.50050    4.283465  0.8826827   3.494715
  0.75025    4.222932  0.8853298   3.439680
  1.00000    4.205678  0.8856945   3.394176

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 1.

$elastic_net_model_1
glmnet

144 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  alpha  lambda  RMSE      Rsquared   MAE
  0.0    0.0     8.735289  0.5994610  7.173798
  0.0    0.4     8.735289  0.5994610  7.173798
  0.0    0.8     9.426153  0.5216627  7.750215
  0.4    0.0     4.583499  0.8630096  3.711615
  0.4    0.4     8.028456  0.6936745  6.677427
  0.4    0.8     9.684120  0.5024980  8.056156
```

```
  0.8    0.0     4.583991  0.8630565  3.711907
  0.8    0.4     7.433367  0.7515882  6.300317
  0.8    0.8     9.884828  0.4911986  8.325876

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.4 and lambda = 0.

$lm_Model_2
Linear Regression

143 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 95, 96, 95
Resampling results:

  RMSE      Rsquared   MAE
  4.189329  0.8986162  3.447526

Tuning parameter 'intercept' was held constant at a value of TRUE

$pls_Model_2
Partial Least Squares

143 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 95, 95
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared   MAE
  1      11.358862  0.2523097  9.043941
  2       7.043822  0.7164522  5.376224
  3       4.905018  0.8606941  3.975397

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 3.

$ridge_Model_2
```

```
Ridge Regression

143 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 95, 95
Resampling results across tuning parameters:

  lambda  RMSE      Rsquared   MAE
  0.000   4.154913  0.9019271  3.444946
  0.025   7.780343  0.7142397  6.248628
  0.050   9.019299  0.6027579  7.157625
  0.075   9.628371  0.5342362  7.595444
  0.100   9.998028  0.4884397  7.850203

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.

$lasso_model_2
The lasso

143 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 95, 95, 96
Resampling results across tuning parameters:

  fraction  RMSE       Rsquared   MAE
  0.00100   12.342615  0.3093525  10.402678
  0.25075    4.320694  0.8963722   3.567534
  0.50050    4.253600  0.9008432   3.515430
  0.75025    4.211394  0.9039716   3.473690
  1.00000    4.194949  0.9057460   3.441437

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 1.

$elastic_net_model_2
glmnet
```

```
143 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 95, 95, 96
Resampling results across tuning parameters:

  alpha  lambda  RMSE      Rsquared   MAE
  0.0    0.0     8.988090  0.5823439  7.177242
  0.0    0.4     8.988090  0.5823439  7.177242
  0.0    0.8     9.144253  0.5636080  7.306462
  0.4    0.0     4.262140  0.8959531  3.569131
  0.4    0.4     7.757239  0.7172453  6.289652
  0.4    0.8     9.456376  0.5250856  7.613519
  0.8    0.0     4.257786  0.8961373  3.565622
  0.8    0.4     7.161422  0.7723034  5.886306
  0.8    0.8     9.696644  0.4947447  7.839422

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.8 and lambda = 0.

$lm_Model_3
Linear Regression

143 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 94
Resampling results:

  RMSE      Rsquared   MAE
  4.506477  0.8760267  3.59666

Tuning parameter 'intercept' was held constant at a value of TRUE

$pls_Model_3
Partial Least Squares

143 samples
```

```
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 94, 96, 96
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared   MAE
  1      10.516071  0.2933451  8.405764
  2       7.700632  0.6335354  6.050398
  3       4.771790  0.8560689  3.773692

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 3.

$ridge_Model_3
Ridge Regression

143 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 95, 95
Resampling results across tuning parameters:

  lambda  RMSE      Rsquared   MAE
  0.000   4.421667  0.8890244  3.539449
  0.025   7.754996  0.6576513  6.126794
  0.050   8.703660  0.5425189  6.862013
  0.075   9.150221  0.4813280  7.205210
  0.100   9.417592  0.4434816  7.401786

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.

$lasso_model_3
The lasso

143 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
```

```
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 95, 95, 96
Resampling results across tuning parameters:

  fraction  RMSE       Rsquared   MAE
  0.00100   11.437704  0.3189084  9.528102
  0.25075    4.866542  0.8889782  3.797613
  0.50050    4.818170  0.8953032  3.758383
  0.75025    4.805276  0.8992952  3.739399
  1.00000    4.828306  0.9009302  3.753759

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.75025.

$elastic_net_model_3
glmnet

143 samples
  5 predictor

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 95, 95
Resampling results across tuning parameters:

  alpha  lambda  RMSE       Rsquared   MAE
  0.0    0.0      8.997280  0.5089241  7.105402
  0.0    0.4      8.997280  0.5089241  7.105402
  0.0    0.8      9.172293  0.4883930  7.243625
  0.4    0.0      4.838175  0.8676164  3.822223
  0.4    0.4      8.155976  0.6130492  6.485942
  0.4    0.8      9.466439  0.4524791  7.546804
  0.8    0.0      4.841118  0.8677438  3.822838
  0.8    0.4      7.657514  0.6675676  6.182328
  0.8    0.8     10.098999  0.3729407  8.113439

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.4 and lambda = 0.
```

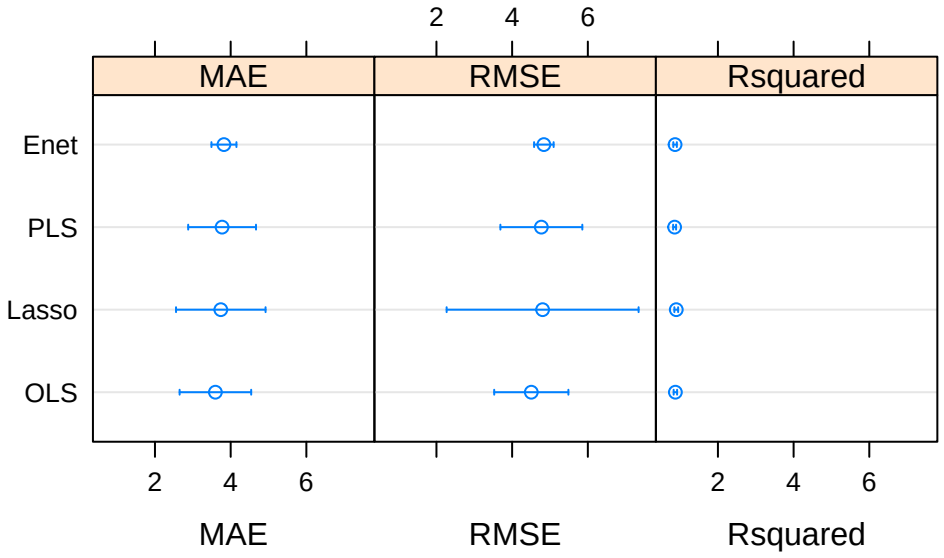
3.1.d (3 points)

Which model has the best predictive ability using RMSE, MAE and R2 (on the training results) as metrics? Is any model significantly better or worse than the others?

```
# Train model evaluation
train_metrics <- resamples(list(OLS = lm_model, PLS = pls_model,
                                Lasso = lasso_model,
                                Enet = elastic_net_model))
summary(train_metrics)
```

```
Call:
summary.resamples(object = train_metrics)

Models: OLS, PLS, Lasso, Enet
Number of resamples: 3

MAE
          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
OLS   3.181297 3.430591 3.679884 3.596660 3.804341 3.928798    0
PLS   3.389824 3.607701 3.825579 3.773692 3.965626 4.105673    0
Lasso 3.416316 3.466032 3.515748 3.739399 3.900941 4.286134    0
Enet  3.696908 3.752541 3.808174 3.822223 3.884881 3.961587    0

RMSE
          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
OLS   4.056022 4.364251 4.672481 4.506477 4.731705 4.790929    0
PLS   4.316738 4.565164 4.813591 4.771790 4.999317 5.185043    0
Lasso 4.031909 4.226679 4.421448 4.805276 5.191960 5.962471    0
Enet  4.774018 4.778314 4.782610 4.838175 4.870254 4.957897    0

Rsquared
           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
OLS   0.8573912 0.8671096 0.8768281 0.8760267 0.8853445 0.8938609    0
PLS   0.8392207 0.8491616 0.8591024 0.8560689 0.8644930 0.8698836    0
Lasso 0.8790515 0.8893344 0.8996172 0.8992952 0.9094170 0.9192168    0
Enet  0.8528422 0.8570748 0.8613074 0.8676164 0.8750035 0.8886995    0
```

```
dotplot(train_metrics)
```



Looking at the averages of the summary output, all models have relatively the same scores for MAE, RMSE, and R^2. When we look at the dotplot of MAE/RMSE/Rsquared, we see a comparable result to the summary() function as all models have relatively the same mean scores illustrated by each circle. However, the distribution is noticeably wider for some models when it comes to MAE and RMSE CL scores. This suggest wider variance. Overall, it looks like Enet has the least variability and have relatively the same mean as the rest of the models.

3.1.e (2 points)

Explain which model you would use for predicting the fat content of a sample.

I would pick Elastic Net model for this instance because the average performance scores across the different models are relatively similar and most of all, it has the least variance amongst the performance metric across the resample iteration.

Problem 3.2 (30 points)

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:

3.2.a Load the data:

```
library(AppliedPredictiveModeling)
data(permeability)
?permeability
permdf <- as.data.frame(cbind(fingerprints, Perm = permeability))
```

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

3.2.b (5 points)

Filter out the predictors that have low frequencies using the `nearZeroVar` function from the caret package. How many predictors are left for modeling after filtering?

```
# Remove low frequency predictors
filtered_permdf <- permdf[, -nearZeroVar(permdf)]
remaining_predictors <- ncol(filtered_permdf) - 1
cat("Predictors left after low frequencies are removed:",
    remaining_predictors)
```

```
Predictors left after low frequencies are removed: 388
```

3.2.c (5 points)

Split the data into a training (80%) and a test set (20%), pre-process the data, and tune a PLS model. How many latent variables are optimal, and what is the corresponding resampled estimate of R2?

```
# Split data into 80% training and 20% test set
set.seed(123)
train_index0 <- createDataPartition(filtered_permdf$permeability, p = 0.8, list = FALSE)
train_data0 <- filtered_permdf[train_index0, ]
test_data0 <- filtered_permdf[-train_index0, ]

# Create PLS model with tuning
ctrl0 <- trainControl(method = "cv", number = 5)
pls_tune <- train(permeability ~ .,
```

```
                    data = train_data0,
                    method = "pls",
                    trControl = ctrl0,
                    tuneGrid = data.frame(ncomp = 1:10),
                    preProcess = c("center", "scale"))

# Determine the number of optimal latent variables
optimal_lv_pls <- pls_tune$bestTune$ncomp
if(is.null(optimal_lv_pls)) {
  print("Error: Unable to determine the optimal number of latent variables.")
} else {
print(paste("Optimal number of latent variables:", optimal_lv_pls))

# Fit the final PLS model using the optimal number of latent variables
pls_model <- caret::train(permeability ~ .,
                          data = train_data0,
                          method = "pls",
                          trControl = ctrl0,
                          preProcess = c("center", "scale"),
                          tuneGrid = expand.grid(ncomp = optimal_lv_pls))

# Evaluate the model performance
R2_pls <- pls_model$results$Rsquared
print(paste("Resampled estimate of R2:", round(R2_pls,4)))}
```

```
[1] "Optimal number of latent variables: 7"
[1] "Resampled estimate of R2: 0.511"
```

3.2.d (3 points)

Predict the response for the test set. What is the test set estimate of R2?

```
# Predict the response for the test set
test_pred_pls <- predict(pls_model, newdata = test_data0)

# Evaluate the model performance
test_R2_pls <- (cor(test_pred_pls, test_data0$permeability))^2
print(paste("Test set estimate of R2:", round(test_R2_pls,4)*100))
```

```
[1] "Test set estimate of R2: 36.18"
```

3.2.e (15 points)

Try building lasso, ridge and elastic net regression models discussed in this chapter. Do any have better predictive performance using R2 as metric on the test data?

```
# Create Ridge Regression Model: Optimized
ridge_grid <- expand.grid(lambda = seq(0.01, 1, length = 10))
ridge_tune <- train(permeability ~ .,
                    data = train_data0,
                    method = "ridge",
                    tuneGrid = ridge_grid,
                    trControl = ctrl0,
                    preProcess = c("center", "scale"))
optimal_lv_lambda <- ridge_tune$bestTune$lambda
ridge_model <- train(permeability ~ .,
                     data = train_data0,
                     method = "ridge",
                     trControl = ctrl0,
                     preProcess = c("center", "scale"),
                     tuneGrid = data.frame(lambda = optimal_lv_lambda))

# Create Lasso Regression Model: Optimized
lasGrid0 <- expand.grid(fraction = seq(0.001, 1, length = 5))
lasso_tune <- train(permeability ~ .,
                    data = train_data0,
                    method = "lasso",
                    tuneGrid = lasGrid0,
                    trControl = ctrl0,
                    preProcess = c("center", "scale"))
optimal_lv_las <- lasso_tune$bestTune$fraction
lasso_model <- train(permeability ~ .,
                    data = train_data0,
                    method = "lasso",
                    trControl = ctrl0,
                    preProcess = c("center", "scale"),
                    tuneGrid = expand.grid(fraction = optimal_lv_las))

#Create Elastic Net Models: Optimized
enetGrid0 <- expand.grid(alpha = seq(0, 1, by = 0.1),
                         lambda = seq(0.1, 1, by = 0.1))
elastic_net_tune <- train(permeability ~ .,
                          data = train_data0,
```

```
                          method = "glmnet",
                          trControl = ctrl0,
                          tuneGrid = enetGrid0,
                          preProcess = c("center", "scale"))
optimal_lv_eneta <- elastic_net_tune$bestTune$alpha
optimal_lv_enetl <- elastic_net_tune$bestTune$lambda
elastic_net_model <- train(permeability ~ .,
                          data = train_data0,
                          method = "glmnet",
                          trControl = ctrl0,
                          preProcess = c("center", "scale"),
                          tuneGrid = expand.grid(alpha = optimal_lv_eneta,
                                                 lambda = optimal_lv_enetl))
```

```
# Run predictions on test data set and determine R^2s
ridge_test_R2 <- cor(predict(ridge_model, newdata = test_data0),
                     test_data0$permeability)^2
lasso_test_R2 <- cor(predict(lasso_model, newdata = test_data0),
                     test_data0$permeability)^2
elastic_net_test_R2 <- cor(predict(elastic_net_model, newdata = test_data0),
                           test_data0$permeability)^2

# Train R^2 scores
R2_ridge <- ridge_model$results$Rsquared
R2_lasso <- lasso_model$results$Rsquared
R2_enet <- elastic_net_model$results$Rsquared

# Table R-squared results for train and test data set
model_results <- data.frame(
  Model = c("PLS","Ridge", "Lasso", "ENet"),
  R2_oTrain = c(round(R2_pls,4)*100,round(R2_ridge,4)*100,
                round(R2_lasso,4)*100,round(R2_enet,4)*100),
  R2_Test = c(round(test_R2_pls,4)*100,round(ridge_test_R2,4)*100,
              round(lasso_test_R2,4)*100, round(elastic_net_test_R2,4)*100))

knitr::kable(model_results, caption = "Test R2 Scores for Different Models")
```

Table 1: Test R2 Scores for Different Models

Model	R2_oTrain	R2_Test
PLS	51.10	36.18

Model	R2_oTrain	R2_Test
Ridge	41.60	40.10
Lasso	33.22	1.90
ENet	52.11	32.67

Yes, Ridge regression model performed the best amongst the 4 models being compared at ~40% r-squared score increase on the test data set.

3.2.f (2 points)Would you recommend any of your models to replace the permeability laboratory experiment?

I'm a little hesitant to recommend because of the low R^2 results.The models fell short in establishing a reliable relationship between the outcome and the predictors. The models passed with test data set performed poorly compared to the optimized model using the train data set. Only Ridge performed relatively the same for both train and test data sets which is good. But the performance is still less than 50% R-squared overall during testing for all models. It's also important to note that 85% of the predictors were removed from the data set during zero variance pre-processing step. This lost of information might have been a contributor to the reduced predictive performance.

Problem 3.3 (30 points)

3.3.a

```
library(AppliedPredictiveModeling)
data(ChemicalManufacturingProcess)
?ChemicalManufacturingProcess
```

The ChemicalManufacturingProcess data frame contains 57 predictors (12 describing the input biological material and 45 describing the process predictors) and a yield column which is the percent yield for each run for the 176 manufacturing runs.

3.3.b (4 points)

Split the data into a training (80%) and a test set (20%). A small percentage of cells in the predictor set contain missing values. Use an imputation function to fill in these missing values in both training and test data sets, also perform centering and scaling.

```r
# Determine each predictor's missing values
ChemicalManufacturingProcess[ChemicalManufacturingProcess == ""] <- NA
na_value <- sapply(ChemicalManufacturingProcess, function(x) sum(is.na(x)))
predictors_with_missing <- names(na_value[na_value > 0])

missing_values_table <- data.frame(Predictor = predictors_with_missing,
                                   Missing_Values = na_value[predictors_with_missing])
missing_values_table |> head() |> gt() |>
  tab_header(title = "Predictors with Missing Values")
```

Predictors with Missing Values

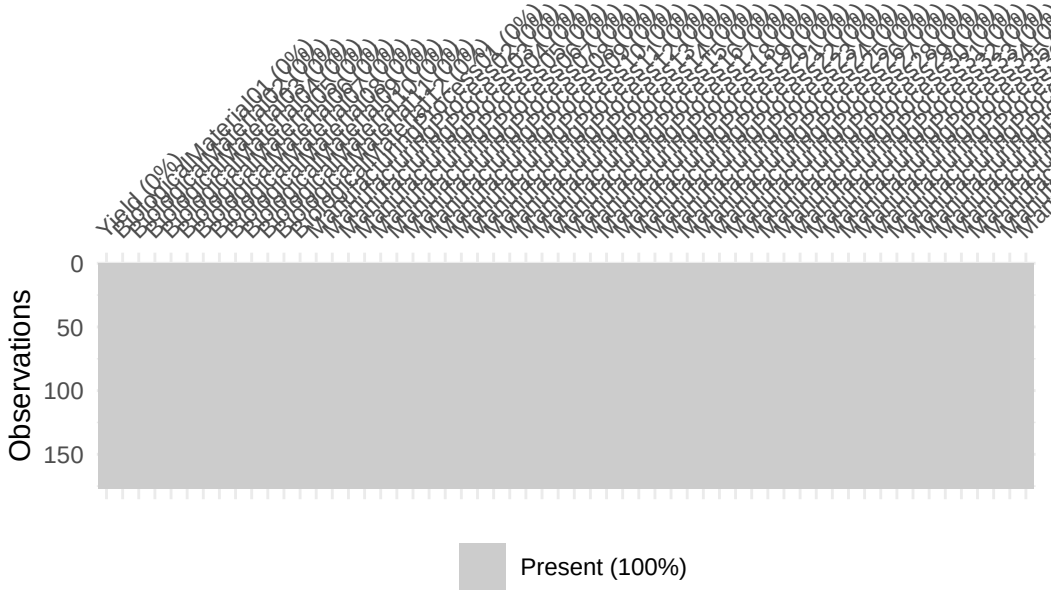
Predictor	Missing_Values
ManufacturingProcess01	1
ManufacturingProcess02	3
ManufacturingProcess03	15
ManufacturingProcess04	1
ManufacturingProcess05	1
ManufacturingProcess06	2

```r
# Imputation by MICE (Multiple Imputation by Chained Equations)
imputed_data <- mice(ChemicalManufacturingProcess)
```

```
 iter imp variable
  1   1  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  1   2  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  1   3  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  1   4  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  1   5  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  2   1  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  2   2  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  2   3  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  2   4  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  2   5  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  3   1  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  3   2  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  3   3  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  3   4  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  3   5  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
```

```
  4   1  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  4   2  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  4   3  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  4   4  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  4   5  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  5   1  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  5   2  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  5   3  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  5   4  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi
  5   5  ManufacturingProcess01  ManufacturingProcess02  ManufacturingProcess03  Manufacturi

Warning: Number of logged events: 675
```

```r
CMfgP_df <- complete(imputed_data)

# Visualize missing values after imputation
vis_miss(CMfgP_df)
```




```r
# Split data into 80% training and 20% test set
set.seed(123)
train_index1 <- createDataPartition(CMfgP_df$Yield, p = 0.8, list = FALSE)
train_data1 <- CMfgP_df[train_index1, ]
test_data1 <- CMfgP_df[-train_index1, ]
```

```
# Perform center and scaling on the train and test data
train_predictors <- subset(train_data1, select = -Yield)
train_outcome <- train_data1$Yield
test_predictors <- subset(test_data1, select = -Yield)
test_outcome <- test_data1$Yield

preproc_train <- preProcess(train_predictors, method = c("center", "scale"))

scaled_train_predictors <- predict(preproc_train, train_predictors)
scaled_test_predictors <- predict(preproc_train, test_predictors)

scaled_train_data <- cbind(Yield = train_outcome, scaled_train_predictors)
scaled_test_data <- cbind(Yield = test_outcome, scaled_test_predictors)
```

3.3.c (16 points)

Tune lasso regression model (lasso), ridge regression model (ridge), partial least squares model (pls), and elastic net model (enet) from chapter 6. What is the optimal value of the resampled performance metric RMSE?

```
set.seed(123)
ctrl1 <- trainControl(method = "cv", number = 5)

# Create PLS Regression Model: Optimized
pls_tune1 <- train(Yield ~ .,
                   data = scaled_train_data,
                   method = "pls",
                   trControl = ctrl1,
                   tuneGrid = data.frame(ncomp = 1:10))
optimal_pls <- pls_tune1$bestTune$ncomp
pls_model1 <- caret::train(Yield ~ .,
                           data = scaled_train_data,
                           method = "pls",
                           trControl = ctrl1,
                           tuneGrid = expand.grid(ncomp = optimal_pls))

# Create Ridge Regression Model: Optimized
ridge_grid1 <- expand.grid(lambda = seq(0.01, 10, length = 10))
ridge_tune1 <- train(Yield ~ .,
                     data = scaled_train_data,
                     method = "ridge",
```

```
                    tuneGrid = ridge_grid1,
                    trControl = ctrl1)
optimal_lambda <- ridge_tune1$bestTune$lambda
ridge_model1 <- train(Yield ~ .,
                     data = scaled_train_data,
                     method = "ridge",
                     trControl = ctrl1,
                     tuneGrid = data.frame(lambda = optimal_lambda))

# Create Lasso Regression Model: Optimized
lasGrid1 <- expand.grid(fraction = seq(0.001, 1, length = 5))
lasso_tune1 <- train(Yield ~ .,
                    data = scaled_train_data,
                    method = "lasso",
                    tuneGrid = lasGrid1,
                    trControl = ctrl1)
optimal_las <- lasso_tune1$bestTune$fraction
lasso_model1 <- train(Yield ~ .,
                    data = scaled_train_data,
                    method = "lasso",
                    trControl = ctrl1,
                    tuneGrid = expand.grid(fraction = optimal_las))

#Create Elastic Net Models: Optimized
enetGrid1 <- expand.grid(alpha = seq(0, 1, by = 0.1),
                         lambda = seq(0.1, 1, by = 0.1))
elastic_net_tune1 <- train(Yield ~ .,
                           data = scaled_train_data,
                           method = "glmnet",
                           trControl = ctrl1,
                           tuneGrid = enetGrid1)
optimal_eneta <- elastic_net_tune1$bestTune$alpha
optimal_enetl <- elastic_net_tune1$bestTune$lambda
elastic_net_model1 <- train(Yield ~ .,
                           data = scaled_train_data,
                           method = "glmnet",
                           trControl = ctrl1,
                           tuneGrid = expand.grid(alpha = optimal_eneta,
                                                  lambda = optimal_enetl))
```

```
# Regression optimal model's RMSE score
pls_optimal_rmse <- pls_model1$results$RMSE
```

```
ridge_optimal_rmse <- ridge_model1$results$RMSE
lasso_optimal_rmse <- lasso_model1$results$RMSE
elastic_net_optimal_rmse <- elastic_net_model1$results$RMSE

# Table RMSE results
model_names <- c("PLS", "Ridge", "Lasso", "Elastic Net")
optimal_RMSE <- c(pls_optimal_rmse, ridge_optimal_rmse,
                  lasso_optimal_rmse, elastic_net_optimal_rmse)
optimal_RMSE_df <- data.frame(Model = model_names,
                              Optimal_Train_RMSE = optimal_RMSE)
optimal_RMSE_df |> gt() |> tab_header(
  title = "RMSE for Each Optimized Model")
```

RMSE for Each Optimized Model

Model	Optimal_Train_RMSE
PLS	1.161879
Ridge	2.779404
Lasso	3.050752
Elastic Net	1.163339

3.3.d (5 points)

Predict the response for the test set using the above trained models. What is the value of the performance metric RMSE, and how does this compare with the resampled performance metric RMSE on the training set?

```
# Perform predictions on test data sets
predictions <- list(
  PLS = predict(pls_model1, newdata = scaled_test_data),
  Ridge = predict(ridge_model1, newdata = scaled_test_data),
  Lasso = predict(lasso_model1, newdata = scaled_test_data),
  Elastic_Net = predict(elastic_net_model1, newdata = scaled_test_data))

# Calculate RMSE on test data set
calc_rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))}
rmse_scores <- sapply(predictions, function(pred)
  calc_rmse(scaled_test_data$Yield, pred))
```

```r
# Table RMSE results
optimal_RMSE_df <- data.frame(Model = model_names,Optimal_Train_RMSE = optimal_RMSE,
                              Test_RMSE = rmse_scores)
optimal_RMSE_df |> gt() |> tab_header(
  title = "RMSE Score for Each Model")
```

RMSE Score for Each Model

Model	Optimal_Train_RMSE	Test_RMSE
PLS	1.161879	1.337178
Ridge	2.779404	1.740040
Lasso	3.050752	1.301250
Elastic Net	1.163339	1.246755

Ridge regression performed better during the test session compared to the train session by >50%. This mean that the model is improving at predicting close to the true values, on average. The Lasso and Elastic Net have relatively similar RMSE between the data sets with small improvement which is good. This means that there's no performance drop. Across the different models, Elastic Net is the top performer at 1.2 RMSE. But overall, the test RMSE performances are all close to each other.

3.3.e (5 points)

In the optimal model, how many biological and process predictors remain (whose coefficient is greater than zero)? What are the top five predictors (use absolute values of coefficients) that have the most impact on the yield?

Assuming optimal model is Elastic Net Regression Model because it scored highest on RMSE test results.

```r
# Extract coefficients from the final model: Elastic Net Model
coef_values <- predict(elastic_net_model1$finalModel, type = "coefficients")
predictor_names <- colnames(scaled_train_data)[!colnames(scaled_train_data) %in% "Yield"]

coef_df <- data.frame(Predictor = c("Intercept", predictor_names),
                      Coefficient = c(coef_values[1], coef_values[-1]))

nonzero_coefs <- coef_df[coef_df$Coefficient != 0, ]
nonzero_coefs <- nonzero_coefs[nonzero_coefs$Predictor != "Intercept", ]
nonzero_coefs_unique <- nonzero_coefs |>
```

```
  group_by(Predictor) |>
  slice(which.max(abs(Coefficient))) |>
  ungroup()

# Count predictors unique manufacturing and biological predictors
manufacturing_process_count <- sum(grepl("ManufacturingProcess",
                                         nonzero_coefs_unique$Predictor))
biological_material_count <- sum(grepl("BiologicalMaterial",
                                       nonzero_coefs_unique$Predictor))

cat("Total count of unique predictors with 'ManufacturingProcess':",
    manufacturing_process_count, "\n")
```

```
Total count of unique predictors with 'ManufacturingProcess': 45
```

```
cat("Total count of unique predictors with 'BiologicalMaterial':",
    biological_material_count, "\n")
```

```
Total count of unique predictors with 'BiologicalMaterial': 12
```

```
# Print top five unique predictors with highest coefficient
nonzero_coefs_unique |> arrange(desc(abs(Coefficient))) |>
  slice(1:5)|> gt() |> tab_header(
  title = "Unique Predictors with Highest Coefficient")
```

Unique Predictors with Highest Coefficient

Predictor	Coefficient
ManufacturingProcess27	-4.501765
ManufacturingProcess29	3.985363
ManufacturingProcess20	-1.729223
ManufacturingProcess18	1.698442
BiologicalMaterial03	1.504406