# Week3, Assignment 1

Dave Hurst, Instructor

```
library(tictoc)
tic('render_time')

library(caret)
# ... add additional libraries here ...
library(tidyverse)
library(scales)
library(glmnet)
library(pls)
library(elasticnet)
library(RANN)
library(gt)

seed <- 123
```

### Problem 3.1 (30 points)

Infrared (IR) spectroscopy technology is used to determine the chemical makeup of a substance. The theory of IR spectroscopy holds that unique molecular structures absorb IR frequencies differently. In practice, a spectrometer fires a series of IR frequencies into a sample material, and the device measures the absorbance of the sample at each individual frequency. This series of measurements creates a spectrum profile which can then be used to determine the chemical makeup of the sample material.

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

```
library(caret)
data(tecator)
# Use ?tecator for details
```

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)

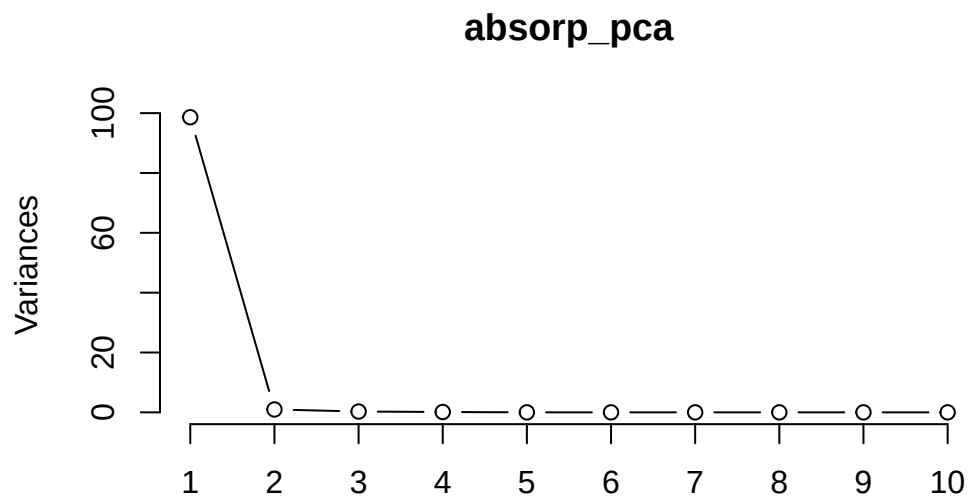
In this example, the predictors are the measurements at the individual frequencies. Because the frequencies lie in a systematic order (850–1,050 nm), the predictors have a high degree of correlation. Hence, the data lie in a smaller dimension than the total number of predictors (100). **Use PCA to determine 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)?**

```
# ... code ...
```

```
# Perform PCA on the absorbance data
absorp_pca <- prcomp(absorp, scale. = TRUE)

# Summarize results to see the cumulative proportion of variance for each PC
summary(absorp_pca)$importance[1:3, 1:5]
```

```
                            PC1       PC2       PC3       PC4        PC5
Standard deviation     9.931072 0.9847361 0.5285114 0.3382748 0.08037979
Proportion of Variance 0.986260 0.0097000 0.0027900 0.0011400 0.00006000
Cumulative Proportion  0.986260 0.9959600 0.9987500 0.9999000 0.99996000
```

```
plot(absorp_pca, type = "l")
```




*... answer ...*

*The first principle component (PC1) explains over 98% of the variance. Since we generally want 95+%, we only need PC1. If we want to capture more of the variance, we could include PC2 which results in a cumulative 99.6% of variance explained.*

### 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)?**

```
# ... code ...
```

```
# Load necessary libraries
#library(caret)
#library(glmnet)  # For ridge, lasso, and elastic net
#library(pls)
library(elasticnet)

# Extract the predictors (absorbance) and response (fat content)
```

```r
absorbance <- as.data.frame(absorp)
fat_content <- endpoints[,2]  #  fat is the second column

# Split the data into training and validation sets
set.seed(seed)  # for reproducibility
train_index <- createDataPartition(fat_content, p = 0.8, list = FALSE)
train_data <- absorbance[train_index, ]
train_fat <- fat_content[train_index]
test_data <- absorbance[-train_index, ]
test_fat <- fat_content[-train_index]

tic('31c')
# Linear Regression
set.seed(seed)  # for reproducibility
lm_model <- train(train_data, train_fat,
                  method = "lm",
                  preProcess = c("center", "scale"),
                  trControl = trainControl(method = "repeatedcv", repeats = 5)
                  )
lm_model$bestTune
```

```
  intercept
1      TRUE
```

```r
# Partial Least Squares
set.seed(seed)  # for reproducibility
pls_model <- train(train_data, train_fat,
                   method = "pls",
                   tuneLength = 40,
                   preProcess = c("center", "scale"),
                   trControl = trainControl(method = "repeatedcv", repeats = 5)
                   )
#plot(pls_model)
pls_model$bestTune
```

```
   ncomp
14    14
```

```r
# Ridge Regression
set.seed(seed)
ridge_grid <- data.frame(.lambda = seq(0, 1, length = 11))
```

```
ridge_model <- train(train_data, train_fat,
                     method = "ridge",
                     tuneGrid =  ridge_grid,
                     preProcess = c("center", "scale"),
                     trControl = trainControl(method = "repeatedcv", repeats = 5)
                     )
#plot(ridge_model)
ridge_model$bestTune
```

```
  lambda
1      0
```

```
# Lasso Regression
set.seed(seed)
lambda_grid <- data.frame(.fraction = seq(0.005, 0.05, length = 20))
lasso_model <- train(train_data, train_fat,
                     method = "lasso",
                     tuneGrid =  lambda_grid,
                     preProcess = c("center", "scale"),
                     trControl = trainControl(method = "repeatedcv", repeats = 5)
)
#plot(lasso_model)
lasso_model$bestTune
```

```
    fraction
7 0.01921053
```

```
# Elastic Net
set.seed(seed)
enet_grid <- expand.grid(.lambda = seq(0, 0.3, length = 4),
                         .fraction = seq(0.005, 0.05, length = 21))
enet_model <- train(train_data, train_fat,
                     method = "enet",
                     tuneGrid =  enet_grid,
                     preProcess = c("center", "scale"),
                     trControl = trainControl(method = "repeatedcv", repeats = 5)
)
#plot(enet_model)
enet_model$bestTune
```

```
  fraction lambda
7   0.0185      0
```

```
toc()
```

```
31c: 157.918 sec elapsed
```

*…Optional recap (required if optimized parameters are not output by code)..*

***Grading notes:***

1. ***Previous key does not create a hold out (test) set, so model metrics differ. Either approach is acceptable since 3.1 does not ask for any test metrics, but in practice one would always evaluate against new data.***
2. ***Lasso and ENET optimal values differ from previous key, because the grid values are adjusted to ensure the optimal fraction does not lie the extreme grid search values or RSME is not sensitive to changing it (i.e. optimized minimum).***

### 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?**

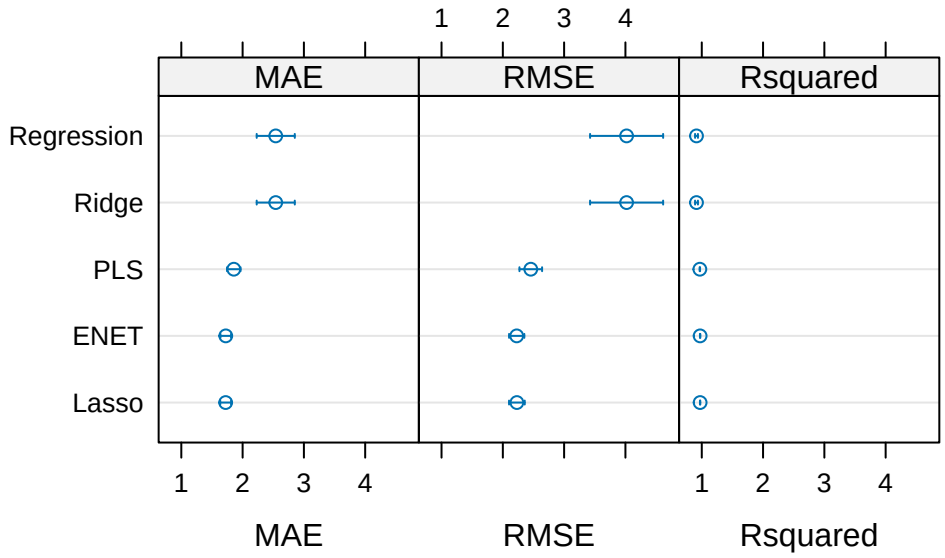
```
# train_metrics <- resamples( ... )
# summary(train_metrics)
# dotplot(train_metrics)
# diff(train_metrics) |> summary()
```

```
train_metrics <- resamples(list(
    Regression = lm_model,
    PLS = pls_model,
    Ridge = ridge_model,
    Lasso = lasso_model,
    ENET = enet_model))
summary(train_metrics)
```

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

```
Models: Regression, PLS, Ridge, Lasso, ENET
Number of resamples: 50

MAE
               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
Regression 1.226077 1.813648 2.213413 2.540577 3.060743 6.803904    0
PLS        1.265008 1.509611 1.777360 1.856492 2.123226 2.855345    0
Ridge      1.226083 1.813592 2.213444 2.540566 3.060780 6.803696    0
Lasso      1.266436 1.518053 1.673106 1.723708 1.885036 2.687690    0
ENET       1.268868 1.518328 1.677596 1.725384 1.885099 2.696296    0

RMSE
               Min.  1st Qu.   Median     Mean  3rd Qu.      Max. NA's
Regression 1.585568 2.432781 3.174158 4.019247 5.035568 11.669293    0
PLS        1.603369 1.995577 2.243776 2.455777 2.768032  4.243505    0
Ridge      1.585543 2.432859 3.174302 4.019254 5.035234 11.668862    0
Lasso      1.489134 1.862771 2.213970 2.229517 2.389937  3.413036    0
ENET       1.483193 1.868737 2.213303 2.226093 2.376637  3.403193    0

Rsquared
                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
Regression 0.6020386 0.8946838 0.9479495 0.9140245 0.9695019 0.9896457    0
PLS        0.9358377 0.9586466 0.9723215 0.9688284 0.9779594 0.9891050    0
Ridge      0.6020560 0.8946851 0.9479484 0.9140242 0.9695010 0.9896461    0
Lasso      0.9508802 0.9686450 0.9747306 0.9731606 0.9795345 0.9903632    0
ENET       0.9516901 0.9685551 0.9749028 0.9732779 0.9794225 0.9905734    0
```

```
dotplot(train_metrics)
```




**Confidence Level: 0.95**

```
diff(train_metrics) |> summary()
```

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

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

MAE
           Regression PLS        Ridge      Lasso      ENET
Regression             6.841e-01  1.058e-05  8.169e-01  8.152e-01
PLS        0.0003971              -6.841e-01  1.328e-01  1.311e-01
Ridge      1.0000000  0.0003971               8.169e-01  8.152e-01
Lasso      1.501e-05  0.0003811  1.501e-05               -1.676e-03
ENET       1.968e-05  0.0004394  1.968e-05  1.0000000

RMSE
           Regression PLS        Ridge      Lasso      ENET
Regression             1.563e+00 -6.838e-06  1.790e+00  1.793e+00
PLS        2.378e-05              -1.563e+00  2.263e-01  2.297e-01
```

```
Ridge       1.000000   2.378e-05              1.790e+00  1.793e+00
Lasso       7.707e-07  0.002158   7.705e-07               3.423e-03
ENET        8.731e-07  0.001434   8.727e-07  1.000000

Rsquared
            Regression PLS        Ridge      Lasso      ENET
Regression             -5.480e-02  3.256e-07 -5.914e-02 -5.925e-02
PLS         0.0002171               5.480e-02 -4.332e-03 -4.450e-03
Ridge       1.0000000  0.0002171              -5.914e-02 -5.925e-02
Lasso       3.973e-05  0.0091552  3.974e-05               -1.174e-04
ENET        4.016e-05  0.0065480  4.016e-05  1.0000000
```

*… Answer …*

*Lasso and ENET have the lowest error (for both RMSE and MSE) and the lowest R2 and their p-values show they are significantly better than the others, but not significantly different from each other. Ridge and linear regression did significantly worse.*

### 3.1.e (2 points)

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

*… Answer …*

*Based on the training results, the Lasso model is arguably best, since it is tied with ENET for the lowest RMSE/MAE and best R2, but has slightly less complexity due to the removal of the Ridge penalty.*

***Grading note: Enet is also an acceptable choice.***

## 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)
# use ?permeability to see more details
```

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)

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 after filtering?**

```
# ... code ...
```

```
degen_vars <- nearZeroVar(fingerprints)
fp_reduced <- fingerprints[ ,-degen_vars]
ans_32b <- ncol(fp_reduced)
```

*... answer ...*

*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?**

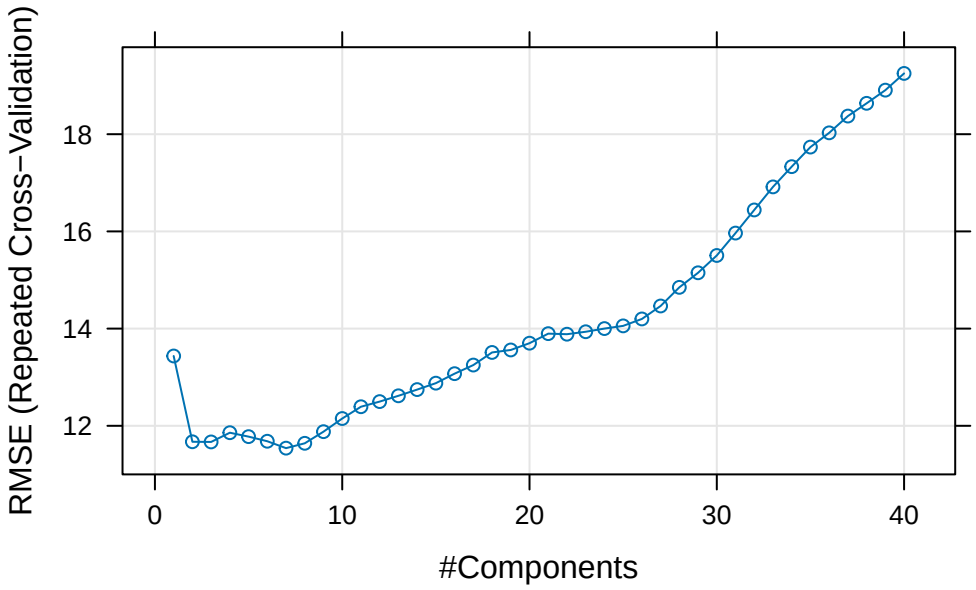
```
# ... code ...
```

```
set.seed(seed)
fp_train_index <- createDataPartition(permeability, p = 0.8, list = FALSE)
train_fingerprints <- fp_reduced[fp_train_index, ]
train_permeability <- permeability[fp_train_index]
test_fingerprints <- fp_reduced[-fp_train_index, ]
test_permeability <- permeability[-fp_train_index]

fp_prep <- preProcess(train_fingerprints, method = c("center", "scale"))
train_fingerprints_trans <- predict(fp_prep, train_fingerprints)
test_fingerprints_trans <- predict(fp_prep, test_fingerprints)

# Partial Least Squares
set.seed(seed)
```

```
fp_pls_model <- train(train_fingerprints_trans, train_permeability,
                      method = "pls",
                      tuneLength = 40,
                      trControl = trainControl(method = "repeatedcv", repeats = 5)
)
fp_pls_model$bestTune
```

```
  ncomp
7     7
```

```
opt_ncomp <- fp_pls_model[["bestTune"]][["ncomp"]]
plot(fp_pls_model)
```




```
opt_R2 <- fp_pls_model$results$Rsquared[opt_ncomp]
opt_R2
```

```
[1] 0.5211716
```

*... Answer ...*

*7 PLS components are optimum, and correspond to an R2 of 0.52*

### 3.2.d (3 points)

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

```
# ... code ...
```

```
fp_pls_predict <- predict(fp_pls_model, newdata = test_fingerprints_trans)
fp_pls_metrics <- postResample(fp_pls_predict, test_permeability)
fp_pls_metrics
```

```
      RMSE   Rsquared        MAE
11.9744490  0.3618213  8.3021291
```

### 3.2.e (15 points)

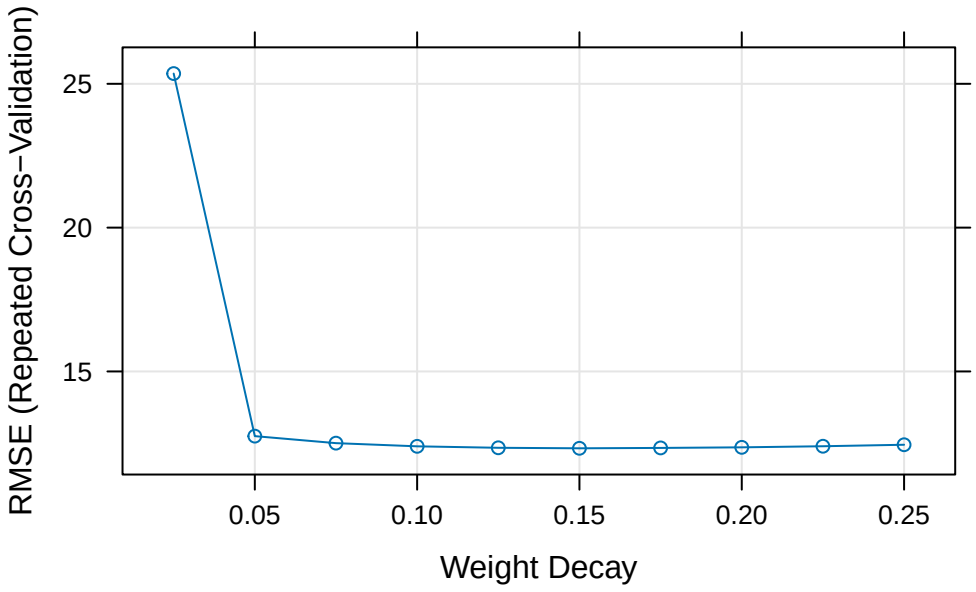
**Try building lasso, ridge and elastic net regression models discussed in this chapter. Do any have better predictive performance using R2 as metric on the test data?**

```
# ... code ...
```

```
# Ridge Regression
set.seed(seed)
tic('ridge')
#ridge_grid <- data.frame(.lambda = seq(0, 0.3, length = 11))
ridge_grid <- data.frame(.lambda = .025 * 1:10)
fp_ridge_model <- train(train_fingerprints_trans, train_permeability,
                        method = "ridge",
                        tuneGrid =  ridge_grid,
                        trControl = trainControl(method = "repeatedcv", repeats = 5)
)
toc()
```

```
ridge: 474.348 sec elapsed
```

```
plot(fp_ridge_model)
```



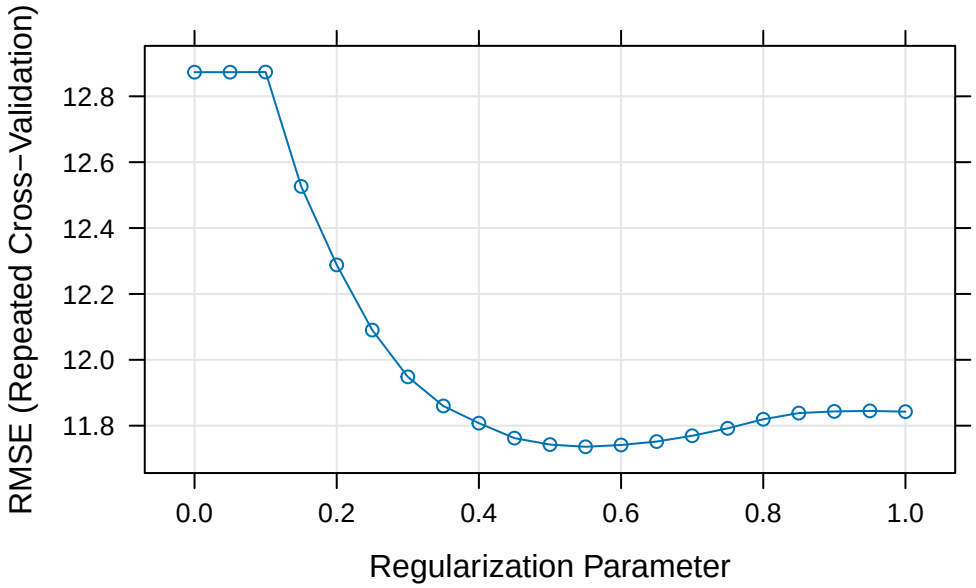

```
fp_ridge_model$bestTune
```

```
  lambda
6   0.15
```

```
# split the chunks so we can work on them separately more efficiently
# due to the long processing times

# Lasso Regression
set.seed(seed)
tic('lasso')
lambda_grid <-  expand.grid(alpha = 1, lambda = seq(0, 1, length = 21))
fp_lasso_model <- train(train_fingerprints_trans, train_permeability,
                    method = "glmnet",  # method = "lasso" is unstable for this dataset
                    tuneGrid =  lambda_grid,
                    trControl = trainControl(method = "repeatedcv", repeats = 5)
)
toc()
```

```
lasso: 3.909 sec elapsed
```

```r
plot(fp_lasso_model)
```




```r
fp_lasso_model$bestTune
```

```
   alpha lambda
12     1   0.55
```

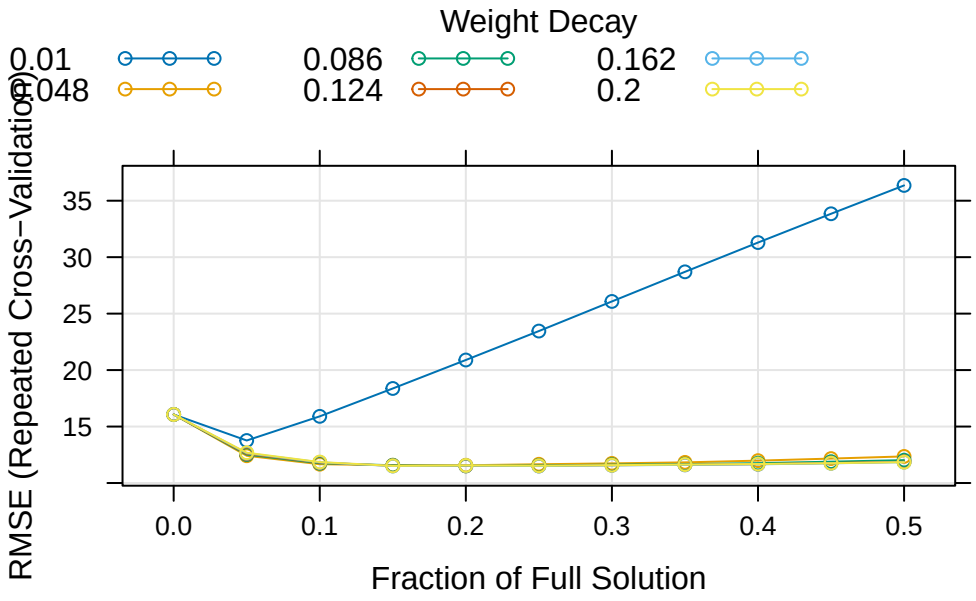
```r
# Elastic Net
set.seed(seed)
tic('enet')
enet_grid <- expand.grid(.lambda = seq(0.01, .2, length = 6),
                         .fraction = seq(0, .5, length = 11))
fp_enet_model <- train(train_fingerprints_trans, train_permeability,
                    method = "enet",
                    tuneGrid =  enet_grid,
                    trControl = trainControl(method = "repeatedcv", repeats = 5)
)
```

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

```
toc()
```

```
enet: 322.971 sec elapsed
```

```
plot(fp_enet_model)
```




```
fp_enet_model$bestTune
```

```
   fraction lambda
50     0.25  0.162
```

```
fp_ridge_predict <- predict(fp_ridge_model, newdata = test_fingerprints_trans)
fp_lasso_predict <- predict(fp_lasso_model, newdata = test_fingerprints_trans)
fp_enet_predict <- predict(fp_enet_model, newdata = test_fingerprints_trans)

fp_ridge_metrics <- postResample(fp_ridge_predict, test_permeability)
fp_lasso_metrics <- postResample(fp_lasso_predict, test_permeability)
fp_enet_metrics <- postResample(fp_enet_predict, test_permeability)

fp_test_r2 <- tribble(
```

```
    ~model, ~Rsquared, ~RMSE,
    'PLS',   fp_pls_metrics['Rsquared'], fp_pls_metrics['RMSE'],
    'Ridge', fp_ridge_metrics['Rsquared'], fp_ridge_metrics['RMSE'],
    'Lasso', fp_lasso_metrics['Rsquared'], fp_lasso_metrics['RMSE'],
    'ENET',  fp_enet_metrics['Rsquared'], fp_enet_metrics['RMSE']
    )
fp_test_r2 |>
    gt::gt()
```

| model | Rsquared | RMSE |
|---|---|---|
| PLS | 0.3618213 | 11.97445 |
| Ridge | 0.4009072 | 12.98318 |
| Lasso | 0.2541716 | 12.71871 |
| ENET | 0.3798061 | 11.88413 |

*...Answer...*

*Ridge regression has the best Rsquared, but simultaneously the worst (highest) RMSE.*

### 3.2.f (2 points)

**Would you recommend any of your models to replace the permeability laboratory experiment?**

*...Answer...*

*The Elasticnet model has the lowest RMSE of any model and has the advantage of more interpretable output, so may be a good alternative to replace the permeability laboratory experiment.*

## Problem 3.3 (30 points)

A chemical manufacturing process for a pharmaceutical product was discussed in Section.1.4 of the textbook. 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:

### 3.3.a

```
library(AppliedPredictiveModeling)
data(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.**

```
# ... code ...
```

```
set.seed(seed)

chem_predictors <- ChemicalManufacturingProcess[ ,-1]
chem_yield <- ChemicalManufacturingProcess[ ,1]

# Split the data into training and test sets
chem_index <- createDataPartition(chem_yield, p = 0.8, list = FALSE)
chem_train <- chem_predictors[chem_index, ]
chem_test <- chem_predictors[-chem_index, ]
yield_train <- chem_yield[chem_index ]
yield_test <- chem_yield[-chem_index ]

# Imputation and scaling using preProcess
chem_prep <- preProcess(chem_train, method = c("center", "scale", "medianImpute"))

# Apply the transformation to the training data
chem_train_transformed <- predict(chem_prep, chem_train)
chem_test_transformed <- predict(chem_prep, chem_test)

na_before <- chem_predictors |> purrr::map_dbl(~ sum(is.na(.x))) |> sum()
na_after <- bind_rows(chem_train_transformed, chem_test_transformed) |>
    purrr::map_dbl(~ sum(is.na(.x))) |> sum()
```

*...how many values were imputed?...*

*106 missing values imputed.*

### 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?**

```
# ... code ...
```

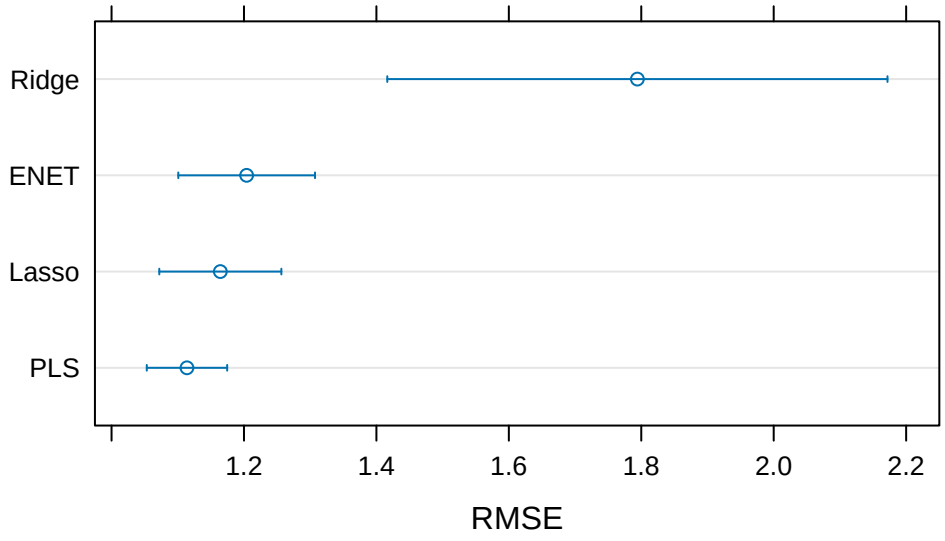
```
# PLS
set.seed(seed)
chem_pls_model <- train(chem_train_transformed, yield_train,
                        method = "pls",
                        tuneLength = 40,
                        trControl = trainControl(method = "repeatedcv", repeats = 5)
)
# plot(chem_pls_model)
# chem_pls_model$bestTune

# Ridge Regression
set.seed(seed)
ridge_grid <- data.frame(.lambda = seq(0, 1, length = 11))
chem_ridge_model <- train(chem_train_transformed, yield_train,
                       method = "ridge",
                       tuneGrid =  ridge_grid,
                       trControl = trainControl(method = "repeatedcv", repeats = 5)
)
# plot(chem_ridge_model)
# chem_ridge_model$bestTune

# Lasso Regression
set.seed(seed)
lambda_grid <- data.frame(.fraction = seq(0.01, .2, length = 20))
#lambda_grid <- data.frame(.fraction = 10^seq(-3, 0, length = 10))
chem_lasso_model <- train(chem_train_transformed, yield_train,
                       method = "lasso",
                       tuneGrid =  lambda_grid,
                       trControl = trainControl(method = "repeatedcv", repeats = 5)
)
```

```r
# plot(chem_lasso_model)
# chem_lasso_model$bestTune


# split the chunks so we can work on them separately more efficiently
# due to the long processing times
# Elastic Net
set.seed(seed)
enet_grid <- expand.grid(.lambda = seq(0.1, .6, length = 6),
                         .fraction = seq(0, .6, length = 11))
chem_enet_model <- train(chem_train_transformed, yield_train,
                       method = "enet",
                       tuneGrid =  enet_grid,
                       trControl = trainControl(method = "repeatedcv", repeats = 5)
)
```

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

```r
# plot(chem_enet_model)
# chem_enet_model$bestTune
```

```r
chem_train_metrics <- resamples(list(
    PLS = chem_pls_model,
    Ridge = chem_ridge_model,
    Lasso = chem_lasso_model,
    ENET = chem_enet_model))
dotplot(chem_train_metrics, metric = 'RMSE')
```




**Confidence Level: 0.95**

```
diff(chem_train_metrics, metric = 'RMSE') |> summary()
```

```
Call:
summary.diff.resamples(object = diff(chem_train_metrics, metric = "RMSE"))

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

RMSE
      PLS      Ridge    Lasso    ENET
PLS           -0.68019 -0.05036 -0.09013
Ridge 0.004993          0.62984  0.59006
Lasso 0.975786 0.006537          -0.03977
ENET  0.384242 0.002222 1.000000
```

*...Answer...*

```
chem_pls_rmse <- summary(chem_train_metrics)$values$`PLS~RMSE` |> mean() |> round(3)
```

*The PLS model performs the best on the resampled training data with an RMSE of 1.114.*

### 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?**

```
# ... code ...
```

```
chem_yhat_pls <- predict(chem_pls_model, newdata = chem_test_transformed)
chem_yhat_ridge <- predict(chem_ridge_model, newdata = chem_test_transformed)
chem_yhat_lasso <- predict(chem_lasso_model, newdata = chem_test_transformed)
chem_yhat_enet <- predict(chem_enet_model, newdata = chem_test_transformed)

chem_pls_metrics <- postResample(chem_yhat_pls, yield_test)
chem_ridge_metrics <- postResample(chem_yhat_ridge, yield_test)
chem_lasso_metrics <- postResample(chem_yhat_lasso, yield_test)
chem_enet_metrics <- postResample(chem_yhat_enet, yield_test)


chem_results <- tibble(
    metric = names(chem_pls_metrics),
    PLS = chem_pls_metrics,
    Ridge = chem_ridge_metrics,
    Lasso = chem_lasso_metrics,
    ENET = chem_enet_metrics
) |> t() |> as_tibble(rownames = 'metric')
```

```
Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
`.name_repair` is omitted as of tibble 2.0.0.
i Using compatibility `.name_repair`.
```

```
chem_results <- tibble(
    Model = c('PLS', 'Ridge', 'Lasso', 'ENET'),
    `Resampled RMSE` = c(
        mean(summary(chem_train_metrics)[["values"]][["PLS~RMSE"]]),
        mean(summary(chem_train_metrics)[["values"]][["Ridge~RMSE"]]),
        mean(summary(chem_train_metrics)[["values"]][["Lasso~RMSE"]]),
        mean(summary(chem_train_metrics)[["values"]][["ENET~RMSE"]])
    ),
    `Test RMSE` = c(
        chem_pls_metrics['RMSE'],
        chem_ridge_metrics['RMSE'],
```

```
        chem_lasso_metrics['RMSE'],
        chem_enet_metrics['RMSE']
    )
)

chem_results |>
    arrange(`Test RMSE`) |>
    gt() |>
    fmt_number( decimals = 3)
```

| Model | Resampled RMSE | Test RMSE |
|---|---|---|
| Lasso | 1.164 | 1.204 |
| ENET | 1.204 | 1.219 |
| PLS | 1.114 | 1.385 |
| Ridge | 1.794 | 1.504 |

### 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?**

```
# ... code ...
```

```
# Extract the final model at the best tuning parameter
best_fraction <- chem_lasso_model$bestTune$fraction
best_model_info <- predict(chem_lasso_model$finalModel, type = "coefficients")
best_l2_index <- which.min(abs(best_model_info$fraction - best_fraction))
best_coeff <- best_model_info$coefficients[best_l2_index, ]
best_coeff_tbl <- tibble(
    predictor = names(best_coeff),
    coefficient = best_coeff,
    )

# What is the number of non-zero coefficients
non_zero_coeff <- best_coeff_tbl |>
    mutate(`Coeff. = 0` = coefficient == 0) |>
    count(`Coeff. = 0`, name = 'Count')
non_zero_coeff |> gt()
```

| Coeff. = 0 | Count |
|---|---|
| FALSE | 7 |
| TRUE | 50 |

```
# What are the top five predictors (use absolute values of coefficients) that have the most i
top5_coeff <- best_coeff_tbl |>
    mutate(abs_coeff = abs(best_coeff)) |>
    arrange(-abs_coeff) |>
    head(5) |>
    select(-abs_coeff) |>
    gt() |>
    fmt_number(decimal = 3) |>
    tab_header('Top 5 predictors (by absolute coefficient value)')
top5_coeff
```

Top 5 predictors (by absolute coefficient value)

| predictor | coefficient |
|---|---|
| ManufacturingProcess32 | 0.719 |
| ManufacturingProcess09 | 0.391 |
| ManufacturingProcess17 | −0.202 |
| BiologicalMaterial06 | 0.100 |
| ManufacturingProcess06 | 0.091 |

***Grading Notes: Depending on preprocessing steps, seeds, etc. the students result may steer them towards ENET, which also an accept choice. Here are results for ENET for comparison, but students only need to provide one***

### ENET (alternative)

```
# Extract the final model at the best tuning parameter
best_fraction <- chem_enet_model$bestTune$fraction
best_model_info <- predict(chem_enet_model$finalModel, type = "coefficients")
best_l2_index <- which.min(abs(best_model_info$fraction - best_fraction))
best_coeff <- best_model_info$coefficients[best_l2_index, ]
best_coeff_tbl <- tibble(
    predictor = names(best_coeff),
    coefficient = best_coeff,
```

```
    )

# What is the number of non-zero coefficients
non_zero_coeff <- best_coeff_tbl |>
    mutate(`Coeff. = 0` = coefficient == 0) |>
    count(`Coeff. = 0`, name = 'Count')
non_zero_coeff |> gt()
```

| Coeff. = 0 | Count |
|---|---|
| FALSE | 25 |
| TRUE | 32 |

```
# What are the top five predictors (use absolute values of coefficients) that have the most i
top5_coeff <- best_coeff_tbl |>
    mutate(abs_coeff = abs(best_coeff)) |>
    arrange(-abs_coeff) |>
    head(5) |>
    select(-abs_coeff) |>
    gt() |>
    fmt_number(decimal = 3) |>
    tab_header('Top 5 predictors (by absolute coefficient value)')
top5_coeff
```

Top 5 predictors (by absolute coefficient value)

| predictor | coefficient |
|---|---|
| ManufacturingProcess32 | 0.610 |
| ManufacturingProcess09 | 0.382 |
| ManufacturingProcess17 | −0.301 |
| ManufacturingProcess13 | −0.225 |
| ManufacturingProcess37 | −0.213 |

```
toc()
```