Homework 4 Peer Assessment

Summer Semester 2021

Background

Selected molecular descriptors from the Dragon chemoinformatics application were used to predict bioconcentration factors for 779 chemicals in order to evaluate QSAR (Quantitative Structure Activity Relationship). This dataset was obtained from the UCI machine learning repository.

The dataset consists of 779 observations of 10 attributes. Below is a brief description of each feature and the response variable (logBCF) in our dataset:

1. *nHM* - number of heavy atoms (integer)
2. *piPC09* - molecular multiple path count (numeric)
3. *PCD* - difference between multiple path count and path count (numeric)
4. *X2Av* - average valence connectivity (numeric)
5. *MLOGP* - Moriguchi octanol-water partition coefficient (numeric)
6. *ON1V* - overall modified Zagreb index by valence vertex degrees (numeric)
7. *N.072* - Frequency of RCO-N< / >N-X=X fragments (integer)
8. *B02[C-N]* - Presence/Absence of C-N atom pairs (binary)
9. *F04[C-O]* - Frequency of C-O atom pairs (integer)
10. *logBCF* - Bioconcentration Factor in log units (numeric)

Note that all predictors with the exception of B02[C-N] are quantitative. For the purpose of this assignment, DO NOT CONVERT B02[C-N] to factor. Leave the data in its original format - numeric in R.

Please load the dataset "Bio_pred" and then split the dataset into a train and test set in a 80:20 ratio. Use the training set to build the models in Questions 1-6. Use the test set to help evaluate model performance in Question 7. Please make sure that you are using R version 3.6.X.

Read Data

```
# Clear variables in memory
rm(list = ls())

# Import the libraries
library(CombMSC)
library(boot)
library(leaps)
library(MASS)
library(glmnet)

# Ensure that the sampling type is correct
RNGkind(sample.kind = "Rejection")

# Set a seed for reproducibility
```

```r
set.seed(100)

# Read data
fullData = read.csv("Bio_pred.csv", header = TRUE)

# Split data for traIning and testing
testRows = sample(nrow(fullData), 0.2 * nrow(fullData))
testData = fullData[testRows, ]
trainData = fullData[-testRows, ]
```

Question 1: Full Model

(a) Fit a standard linear regression with the variable *logBCF* as the response and the other variables as predictors. Call it *model1*. Display the model summary.

```r
model1 <- lm(logBCF ~ ., data = trainData)
summary(model1)
```

```
##
## Call:
## lm(formula = logBCF ~ ., data = trainData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2577 -0.5180  0.0448  0.5117  4.0423
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.001422   0.138057   0.010  0.99179
## nHM          0.137022   0.022462   6.100 1.88e-09 ***
## piPC09       0.031158   0.020874   1.493  0.13603
## PCD          0.055655   0.063874   0.871  0.38391
## X2Av        -0.031890   0.253574  -0.126  0.89996
## MLOGP        0.506088   0.034211  14.793  < 2e-16 ***
## ON1V         0.140595   0.066810   2.104  0.03575 *
## N.072       -0.073334   0.070993  -1.033  0.30202
## B02.C.N.    -0.158231   0.080143  -1.974  0.04879 *
## F04.C.O.    -0.030763   0.009667  -3.182  0.00154 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7957 on 614 degrees of freedom
## Multiple R-squared:  0.6672, Adjusted R-squared:  0.6623
## F-statistic: 136.8 on 9 and 614 DF,  p-value: < 2.2e-16
```

(b) Which regression coefficients are significant at the 95% confidence level? At the 99% confidence level?
At the 95% confidence interval: nHM, MLOGP, ONV1, BO2.C.N., F04.C.0.
At the 99% confidence interval: nHM, MLOGP, F04.C.O.

(c) What are the 10-fold and leave one out cross-validation scores for this model?

```r
set.seed(100)

model1_cv = glm(logBCF ~ ., data = trainData)

# 10-fold
ten_fold <- cv.glm(trainData, model1_cv, K = 10)
print(ten_fold$delta[1])
```

```
## [1] 0.6512928
```

```r
# Leave One Out
leave_one <- cv.glm(trainData, model1_cv, K = nrow(trainData))
print(leave_one$delta[1])
```

```
## [1] 0.6529872
```

(d) What are the Mallow's Cp, AIC, and BIC criterion values for this model?

```r
set.seed(100)
mallow_cp <- Cp(model1, S2 = summary(model1)$sigma^2)
aic <- AIC(model1, k = 2)
bic <- AIC(model1, k = log(nrow(trainData)))

print(paste0("Mallow's CP: ", round(mallow_cp, 0)))
```

```
## [1] "Mallow's CP: 10"
```

```r
print(paste0("AIC: ", round(aic, 0)))
```

```
## [1] "AIC: 1497"
```

```r
print(paste0("BIC: ", round(bic, 0)))
```

```
## [1] "BIC: 1546"
```

(e) Build a new model on the training data with only the variables which coefficients were found to be statistically significant at the 99% confident level. Call it *model2*. Perform an ANOVA test to compare this new model with the full model. Which one would you prefer? Is it good practice to select variables based on statistical significance of individual coefficients? Explain.

```r
set.seed(100)
model2 <- lm(logBCF ~ nHM + MLOGP + F04.C.O., data = trainData)

anova(model2, model1)
```

```
## Analysis of Variance Table
##
## Model 1: logBCF ~ nHM + MLOGP + F04.C.O.
## Model 2: logBCF ~ nHM + piPC09 + PCD + X2Av + MLOGP + ON1V + N.072 + B02.C.N. +
```

```
##     F04.C.O.
##   Res.Df    RSS Df Sum of Sq     F  Pr(>F)
## 1    620 400.51
## 2    614 388.70  6    11.809 3.109 0.00523 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

P value of 0.00523 is close to 0 and indicates that we reject the null hypothesis that all the coefficients in the full model are equal to 0. Based on this, we should choose the full model over the reduced one. However, it is not good practice to select predictor variables using this methodology.

Question 2: Full Model Search

(a) Compare all possible models using Mallow's Cp. What is the total number of possible models with the full set of variables? Display a table indicating the variables included in the best model of each size and the corresponding Mallow's Cp value.

Hint: You can use nbest parameter.

```
set.seed(100)

mallow_cp_full <- leaps(trainData[, -10], trainData$logBCF, method = "Cp", nbest = 1,
    names = names(trainData)[-10])

cbind(as.matrix(mallow_cp_full$which), mallow_cp_full$Cp)
```

```
##   nHM piPC09 PCD X2Av MLOGP ON1V N.072 B02.C.N. F04.C.O.
## 1   0      0   0    0     1    0     0        0        0 58.596851
## 2   1      0   0    0     1    0     0        0        0 17.737801
## 3   1      1   0    0     1    0     0        0        0 15.184626
## 4   1      1   0    0     1    0     0        0        1  9.495041
## 5   1      1   0    0     1    0     0        1        1  7.240754
## 6   1      1   0    0     1    1     0        1        1  6.116174
## 7   1      1   0    0     1    1     1        1        1  6.831852
## 8   1      1   1    0     1    1     1        1        1  8.015816
## 9   1      1   1    1     1    1     1        1        1 10.000000
```

Total number of models $= 2^9 = 512$.

(b) How many variables are in the model with the lowest Mallow's Cp value? Which variables are they? Fit this model and call it *model3*. Display the model summary.

```
set.seed(100)
model3 <- lm(logBCF ~ nHM + piPC09 + MLOGP + ON1V + B02.C.N. + F04.C.O., data = trainData)
summary(model3)
```

```
##
## Call:
## lm(formula = logBCF ~ nHM + piPC09 + MLOGP + ON1V + B02.C.N. +
##     F04.C.O., data = trainData)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2364 -0.5234  0.0421  0.5196  4.1159
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.035785   0.099454   0.360  0.71911
## nHM          0.124086   0.019083   6.502 1.63e-10 ***
## piPC09       0.042167   0.014135   2.983  0.00297 **
## MLOGP        0.528522   0.029434  17.956  < 2e-16 ***
## ON1V         0.098099   0.055457   1.769  0.07740 .
## B02.C.N.    -0.160204   0.073225  -2.188  0.02906 *
## F04.C.O.    -0.028644   0.009415  -3.042  0.00245 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7951 on 617 degrees of freedom
## Multiple R-squared:  0.666,  Adjusted R-squared:  0.6628
## F-statistic: 205.1 on 6 and 617 DF,  p-value: < 2.2e-16
```

There are 6 variables in the model with the lowest Mallow's Cp value. This model includes nHM, piPC09, MLOGP, ON1V, B02.C.N., and F04.C.O.

Question 3: Stepwise Regression

(a) Perform backward stepwise regression using BIC. Allow the minimum model to be the model with only an intercept, and the full model to be *model1*. Display the model summary of your final model. Call it *model4*

```
set.seed(100)

min_model <- lm(logBCF ~ 1, data = trainData)
model4 = step(model1, scope = list(lower = min_model, upper = model1), direction = "backward",
    k = log(nrow(trainData)), trace = "F")
summary(model4)
```

```
##
## Call:
## lm(formula = logBCF ~ nHM + piPC09 + MLOGP + F04.C.O., data = trainData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2611 -0.5126  0.0517  0.5353  4.3488
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.008695   0.078196  -0.111  0.91150
## nHM          0.114029   0.017574   6.489 1.78e-10 ***
## piPC09       0.041119   0.013636   3.015  0.00267 **
## MLOGP        0.566473   0.025990  21.796  < 2e-16 ***
## F04.C.O.    -0.022104   0.008000  -2.763  0.00590 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7985 on 619 degrees of freedom
## Multiple R-squared:  0.662,  Adjusted R-squared:  0.6599
## F-statistic: 303.1 on 4 and 619 DF,  p-value: < 2.2e-16
```

(b) How many variables are in *model4*? Which regression coefficients are significant at the 99% confidence level?

There are 4 predictor variables in model 4; all 4 predictors are significant at the 99% confidence level.

(c) Perform forward stepwise selection with AIC. Allow the minimum model to be the model with only an intercept, and the full model to be *model1*. Display the model summary of your final model. Call it *model5*. Do the variables included in *model5* differ from the variables in *model4*?

```
set.seed(100)

model5 = step(min_model, scope = list(lower = min_model, upper = model1), direction = "forward",
    k = 2, trace = F)
summary(model5)
```

```
##
## Call:
## lm(formula = logBCF ~ MLOGP + nHM + piPC09 + F04.C.O. + B02.C.N. +
##     ON1V, data = trainData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2364 -0.5234  0.0421  0.5196  4.1159
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.035785   0.099454   0.360  0.71911
## MLOGP        0.528522   0.029434  17.956  < 2e-16 ***
## nHM          0.124086   0.019083   6.502 1.63e-10 ***
## piPC09       0.042167   0.014135   2.983  0.00297 **
## F04.C.O.    -0.028644   0.009415  -3.042  0.00245 **
## B02.C.N.    -0.160204   0.073225  -2.188  0.02906 *
## ON1V         0.098099   0.055457   1.769  0.07740 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7951 on 617 degrees of freedom
## Multiple R-squared:  0.666,  Adjusted R-squared:  0.6628
## F-statistic: 205.1 on 6 and 617 DF,  p-value: < 2.2e-16
```

Yes, the variables chosen for model5 differ than those chosen for model4. There are 6 predictor variables, with B02.C.N and ON1V in addition to the ones chosen in model4.

(d) Compare the adjusted R^2, Mallow's Cp, AICs and BICs of the full model(*model1*), the model found in Question 2 (*model3*), and the model found using backward selection with BIC (*model4*). Which model is preferred based on these criteria and why?

```r
set.seed(100)

print("Adj. R Squared")
```

```
## [1] "Adj. R Squared"
```

```r
print(paste0("Full Model: ", round(summary(model1)$adj.r.sq, 2)))
```

```
## [1] "Full Model: 0.66"
```

```r
print(paste0("Complete Search with Mallow's Cp: ", round(summary(model3)$adj.r.sq,
    2)))
```

```
## [1] "Complete Search with Mallow's Cp: 0.66"
```

```r
print(paste0("Backward Selection with BIC: ", round(summary(model4)$adj.r.sq, 2)))
```

```
## [1] "Backward Selection with BIC: 0.66"
```

```r
library(caret)
print("Mallow's CP")
```

```
## [1] "Mallow's CP"
```

```r
print(paste0("Full Model: ", round(Cp(model1, S2 = summary(model1)$sigma^2)[1], 2)))
```

```
## [1] "Full Model: 10"
```

```r
print(paste0("Complete Search with Mallow's Cp: ", round(Cp(model3, S2 = summary(model3)$sigma^2)[1],
    2)))
```

```
## [1] "Complete Search with Mallow's Cp: 7"
```

```r
print(paste0("Backward Selection with BIC: ", round(Cp(model4, S2 = summary(model4)$sigma^2)[1],
    2)))
```

```
## [1] "Backward Selection with BIC: 5"
```

```r
print("AIC")
```

```
## [1] "AIC"
```

```r
print(paste0("Full Model: ", round(AIC(model1, k = 2), 2)))
```

```
## [1] "Full Model: 1497.48"
```

```
print(paste0("Complete Search with Mallow's Cp: ", round(AIC(model3, k = 2), 2)))
```

```
## [1] "Complete Search with Mallow's Cp: 1493.62"
```

```
print(paste0("Backward Selection with BIC: ", round(AIC(model4, k = 2), 2)))
```

```
## [1] "Backward Selection with BIC: 1497.05"
```

```
print("BIC")
```

```
## [1] "BIC"
```

```
print(paste0("Full Model: ", round(AIC(model1, k = log(nrow(trainData))), 2)))
```

```
## [1] "Full Model: 1546.27"
```

```
print(paste0("Complete Search with Mallow's Cp: ", round(AIC(model3, k = log(nrow(trainData))),
    2)))
```

```
## [1] "Complete Search with Mallow's Cp: 1529.11"
```

```
print(paste0("Backward Selection with BIC: ", round(AIC(model4, k = log(nrow(trainData))),
    2)))
```

```
## [1] "Backward Selection with BIC: 1523.67"
```

All three models have the same adjusted r squared. Based on the Mallow's CP and BIC, the preferred model is model 4, Backwards Selection with BIC, because it has the lowest values.

Question 4: Ridge Regression

(a) Perform ridge regression on the training set. Use cv.glmnet() to find the lambda value that minimizes the cross-validation error using 10 fold CV.

```
set.seed(100)
train_x <- as.matrix(trainData[, -10])
train_y <- trainData[, 10]
ridge_reg <- cv.glmnet(train_x, train_y, family = "gaussian", alpha = 0, nfolds = 10)

ridge_reg$lambda.min
```

```
## [1] 0.108775
```

Lambda value of 0.11 minimizes the cross-validation error.

(b) List the value of coefficients at the optimum lambda value.

```
set.seed
```

```
## function (seed, kind = NULL, normal.kind = NULL, sample.kind = NULL)
## {
##     kinds <- c("Wichmann-Hill", "Marsaglia-Multicarry", "Super-Duper",
##         "Mersenne-Twister", "Knuth-TAOCP", "user-supplied", "Knuth-TAOCP-2002",
##         "L'Ecuyer-CMRG", "default")
##     n.kinds <- c("Buggy Kinderman-Ramage", "Ahrens-Dieter", "Box-Muller",
##         "user-supplied", "Inversion", "Kinderman-Ramage", "default")
##     s.kinds <- c("Rounding", "Rejection", "default")
##     if (length(kind)) {
##         if (!is.character(kind) || length(kind) > 1L)
##             stop("'kind' must be a character string of length 1 (RNG to be used).")
##         if (is.na(i.knd <- pmatch(kind, kinds) - 1L))
##             stop(gettextf("'%s' is not a valid abbreviation of an RNG",
##                 kind), domain = NA)
##         if (i.knd == length(kinds) - 1L)
##             i.knd <- -1L
##     }
##     else i.knd <- NULL
##     if (!is.null(normal.kind)) {
##         if (!is.character(normal.kind) || length(normal.kind) !=
##             1L)
##             stop("'normal.kind' must be a character string of length 1")
##         normal.kind <- pmatch(normal.kind, n.kinds) - 1L
##         if (is.na(normal.kind))
##             stop(gettextf("'%s' is not a valid choice", normal.kind),
##                 domain = NA)
##         if (normal.kind == 0L)
##             stop("buggy version of Kinderman-Ramage generator is not allowed",
##                 domain = NA)
##         if (normal.kind == length(n.kinds) - 1L)
##             normal.kind <- -1L
##     }
##     if (!is.null(sample.kind)) {
##         if (!is.character(sample.kind) || length(sample.kind) !=
##             1L)
##             stop("'sample.kind' must be a character string of length 1")
##         sample.kind <- pmatch(sample.kind, s.kinds) - 1L
##         if (is.na(sample.kind))
##             stop(gettextf("'%s' is not a valid choice", sample.kind),
##                 domain = NA)
##         if (sample.kind == 0L)
##             warning("non-uniform 'Rounding' sampler used", domain = NA)
##         if (sample.kind == length(s.kinds) - 1L)
##             sample.kind <- -1L
##     }
##     .Internal(set.seed(seed, i.knd, normal.kind, sample.kind))
## }
## <bytecode: 0x7fda1d7cd350>
## <environment: namespace:base>
```

```
ridge_reg_model = cv.glmnet(train_x, train_y, family = "gaussian", alpha = 0, nlambda = 100)
coef(ridge_reg_model, s = ridge_reg$lambda.min)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  0.13841426
## nHM          0.14391877
## piPC09       0.03735762
## PCD          0.08235334
## X2Av        -0.06901352
## MLOGP        0.44403654
## ON1V         0.15770114
## N.072       -0.09683534
## B02.C.N.    -0.20919397
## F04.C.O.    -0.03177144
```

(c) How many variables were selected? Give an explanation for this number.

All 9 predictor variables were selected as ridge regression only shrinks each coefficient but does not force them to 0.

Question 5: Lasso Regression

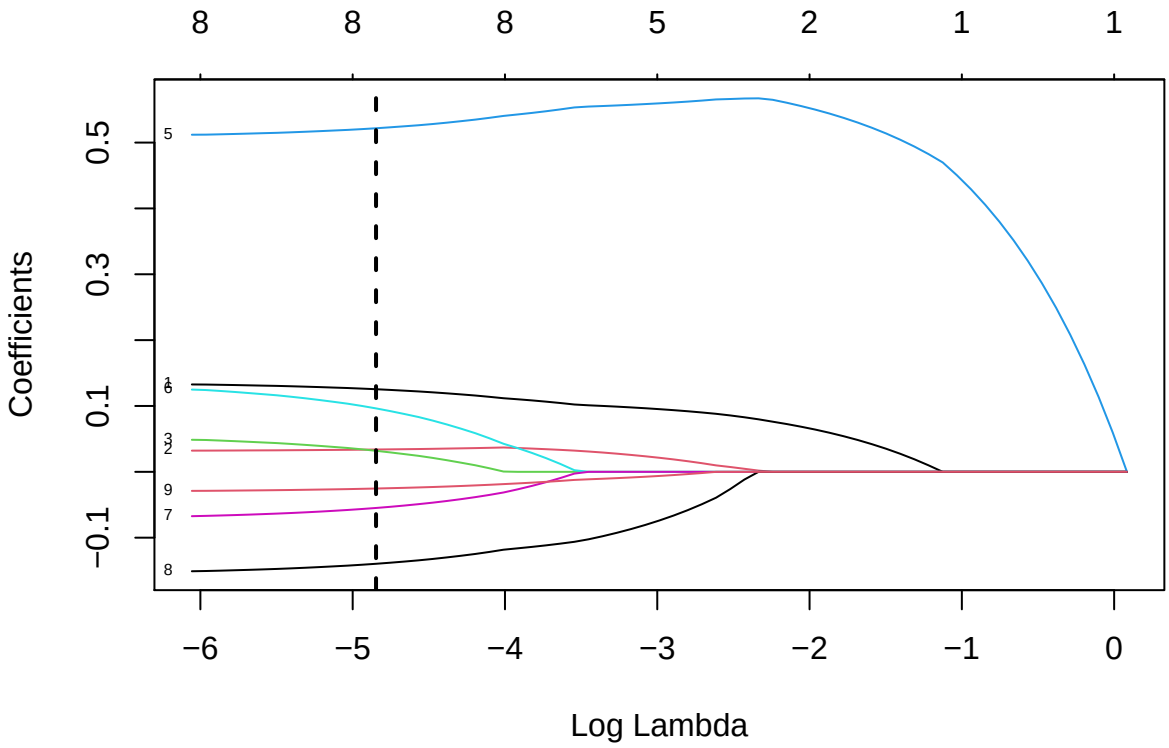
(a) Perform lasso regression on the training set.Use cv.glmnet() to find the lambda value that minimizes the cross-validation error using 10 fold CV.

```
set.seed(100)

lasso <- cv.glmnet(train_x, train_y, family = "gaussian", alpha = 1, nfolds = 10)
lasso$lambda.min
```

```
## [1] 0.007854436
```

(b) Plot the regression coefficient path.

```
set.seed(100)
reg_coef_path <- glmnet(train_x, train_y, family = "gaussian", alpha = 1)

plot(reg_coef_path, xvar = "lambda", label = TRUE)
abline(v = log(lasso$lambda.min), col = "black", lty = 2, lwd = 2)
```



(c) How many variables were selected? Which are they?

```
set.seed(100)
coef(reg_coef_path, s = lasso$lambda.min)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  0.02722838
## nHM          0.12543866
## piPC09       0.03387665
## PCD          0.03194878
## X2Av         .
## MLOGP        0.52174346
## ON1V         0.09633951
## N.072       -0.05487196
## B02.C.N.    -0.13961811
## F04.C.O.    -0.02535576
```

All variables except X2Av were selected using LASSO.

Question 6: Elastic Net

(a) Perform elastic net regression on the training set. Use cv.glmnet() to find the lambda value that minimizes the cross-validation error using 10 fold CV. Give equal weight to both penalties.

```
set.seed(100)

elastic_net <- cv.glmnet(train_x, train_y, family = "gaussian", alpha = 0.5, nfolds = 10)
elastic_net_model <- glmnet(train_x, train_y, family = "gaussian", alpha = 0.5)
elastic_net$lambda.min
```

```
## [1] 0.0207662
```

(b) List the coefficient values at the optimal lambda. How many variables were selected? How do these variables compare to those from Lasso in Question 5?

```
set.seed(100)
coef(elastic_net_model, s = elastic_net$lambda.min)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  0.04903516
## nHM          0.12397290
## piPC09       0.03470891
## PCD          0.03060034
## X2Av         .
## MLOGP        0.51776470
## ON1V         0.08901088
## N.072       -0.05236840
## B02.C.N.    -0.14155538
## F04.C.O.    -0.02420217
```

Similar to LASSO, all the variables were chosen except X2Av.

Question 7: Model comparison

(a) Predict *logBCF* for each of the rows in the test data using the full model, and the models found using backward stepwise regression with BIC, ridge regression, lasso regression, and elastic net.

```
set.seed(100)

full_res <- predict(model1, testData)
back_step <- predict(model4, testData)

test_x = as.matrix(testData[, -10])
ridge_res = as.vector(predict(ridge_reg_model, test_x, s = ridge_reg$lambda.min))
lasso_res = as.vector(predict(reg_coef_path, test_x, s = lasso$lambda.min))
elastic_res = as.vector(predict(elastic_net_model, test_x, s = elastic_net$lambda.min))

final_pred <- data.frame(logBCF = testData$logBCF, full_res, back_step, ridge_res,
    lasso_res, elastic_res)
final_pred
```

```
##     logBCF   full_res  back_step   ridge_res  lasso_res elastic_res
## 714   2.64  2.4464790  2.4249160  2.45487772  2.4428951   2.4415065
```

```
## 503   4.58  4.3337590  4.3531671  4.23442451  4.3135091   4.2964508
## 358   3.44  3.2668917  3.2741920  3.22316616  3.2606171   3.2526382
## 624   0.67  1.6647698  1.2971754  1.73409396  1.6100055   1.6067741
## 718   2.61  1.9553625  2.0013985  1.99396666  1.9399459   1.9394645
## 470   4.81  4.3332777  4.3554698  4.23322240  4.3141282   4.2971705
## 516   0.90  1.1733488  1.2318810  1.16886818  1.1787086   1.1799112
## 98    0.99  0.7920842  0.9745210  0.77770161  0.8738189   0.8925199
## 7     0.83  0.9760207  1.0449449  0.98826194  0.9856630   0.9897446
## 183  -0.16  0.8965352  0.7565987  0.96335645  0.8860562   0.8913939
## 299   1.20  1.6415059  1.5920838  1.66183832  1.6097949   1.6042099
## 504   2.05  2.4289843  2.4657563  2.43150860  2.4144880   2.4105754
## 466   2.05  1.3434192  1.3861348  1.38502737  1.3680733   1.3785079
## 307  -0.30  1.3714599  1.4998130  1.36996049  1.3989513   1.4030411
## 456   1.38  1.6539631  1.7658354  1.62098565  1.6660478   1.6656148
## 146   2.25  2.6016885  2.8094255  2.49773893  2.6390294   2.6364415
## 258   2.07  1.8283856  1.8351099  1.89827140  1.7981214   1.7986375
## 435   5.81  4.5951178  4.6110039  4.48799975  4.5687148   4.5495085
## 324   0.60  0.8765335  1.0605983  0.85643654  0.9313164   0.9435283
## 68    0.79  0.8350958  0.7681078  0.92739563  0.8358476   0.8470279
## 510   1.34  0.8389724  0.9949593  0.83146836  0.9030185   0.9181841
## 560   3.99  2.1043019  2.0153064  2.05629487  2.1343183   2.1348914
## 288   2.12  1.8431284  1.8132090  1.85224416  1.8457291   1.8475949
## 341   4.39  4.0709051  4.0998946  3.97738072  4.0591933   4.0444639
## 347   3.12  2.2735008  2.4039879  2.18083212  2.3355247   2.3370279
## 167   0.34  1.5572615  1.6535542  1.52714033  1.5768856   1.5782385
## 377   4.70  4.3338342  4.3554698  4.23404594  4.3144477   4.2974765
## 628   0.62  1.5705585  1.5966282  1.57583628  1.5988243   1.6049427
## 450   2.30  1.6570691  1.5399971  1.63719292  1.6581150   1.6559618
## 605   0.84  1.2079377  1.2503448  1.23853296  1.2411663   1.2517742
## 301   1.56  0.9944331  1.0210444  1.08462202  0.9960823   1.0065168
## 670   2.18  1.8756853  1.9088333  1.93243376  1.8545452   1.8550857
## 158   3.19  1.9667661  1.8646257  2.07177681  1.9395982   1.9446763
## 733   0.72  0.4344742  0.4185497  0.57924507  0.4138505   0.4266109
## 87    0.16  1.6532873  1.5853651  1.67676727  1.6659880   1.6728686
## 607   0.86  1.5993823  1.6128876  1.59934367  1.6222540   1.6269543
## 223   1.43  1.1114233  1.1347931  1.18201434  1.0961912   1.1025544
## 732  -0.50  1.6939373  1.6484121  1.76492402  1.6941759   1.7019619
## 251   2.43  2.8593328  2.8530648  2.84138006  2.8546043   2.8499279
## 543   3.36  3.0389691  3.0701933  2.95097765  3.0423949   3.0338343
## 694   2.05  1.7442754  1.8926801  1.69242411  1.7750475   1.7770732
## 425  -0.06  0.2816258  0.1676462  0.42175437  0.2761840   0.2912411
## 489   2.16  2.4999765  2.4759516  2.47520026  2.4913611   2.4878859
## 297   2.25  2.0151512  2.0374751  2.07238901  1.9879597   1.9871922
## 502   4.08  3.7551642  3.7329564  3.71367840  3.7310647   3.7202557
## 171   0.69  1.9093530  1.8792628  1.88164223  1.9546481   1.9636395
## 519   2.80  2.3848930  2.3619222  2.35569014  2.3810173   2.3778487
## 703   0.55  0.8930405  0.9586715  0.97581595  0.8968677   0.9064106
## 449   0.86  0.7566587  0.8437902  0.78184119  0.7837119   0.7938474
## 393   1.93  3.0483432  2.9727672  2.93542266  3.0388090   3.0223835
## 660   1.60  2.0653791  1.9902781  2.12554877  2.0700501   2.0773651
## 363   1.81  2.1065283  2.1464387  2.11435996  2.0990734   2.0974659
## 600   1.22  0.9544088  0.9897671  0.98941736  1.0011643   1.0136025
## 387   2.83  2.6249809  2.5567664  2.68257042  2.6035334   2.6046803
## 420   1.57  1.5129871  1.2840851  1.60791732  1.4672374   1.4681792
```

```
## 371   4.26  3.7829499  3.9814591  3.54986489  3.8571331   3.8438148
## 430   2.50  2.4096713  2.4137883  2.39772257  2.4084231   2.4069586
## 254   2.84  2.0139994  2.0769700  2.01785719  2.0334717   2.0380624
## 47    2.00  2.5423003  2.5473264  2.49362419  2.5220487   2.5107569
## 439   2.84  1.9453001  1.9741325  1.91376315  1.9375532   1.9324343
## 708   2.49  2.3631197  2.3345659  2.38890660  2.3574348   2.3574048
## 12    0.31  1.5504093  1.5168886  1.57910355  1.5210294   1.5175218
## 121   0.96  1.1609621  1.2318810  1.15384483  1.1706869   1.1724564
## 16    2.57  2.1375892  2.1039493  2.17186853  2.1374397   2.1401176
## 406   3.50  2.8425947  2.7944007  2.85895099  2.8301348   2.8287450
## 643   2.12  2.6656626  2.5824084  2.71241804  2.6447210   2.6445054
## 133  -0.29 -0.1261240 -0.2749377  0.04680632 -0.1380293  -0.1204352
## 556   0.69  2.1492280  2.1189374  2.18258417  2.1316611   2.1318635
## 156   3.48  1.8244579  1.8783827  1.83462993  1.8352661   1.8399038
## 281   1.98  1.5878690  1.5295821  1.62504943  1.5859192   1.5911117
## 554   1.65  2.3338251  2.4771205  2.26328215  2.3607525   2.3586905
## 655   3.18  2.9945931  2.9124755  3.01221326  2.9608449   2.9538617
## 185   3.30  3.1198641  2.9157760  2.98238799  3.0901223   3.0650680
## 298   2.16  1.2352854  1.0699493  1.36847024  1.2226026   1.2348211
## 421   2.71  2.4201252  2.3854561  2.43229020  2.4429402   2.4494153
## 666   1.01  2.2401644  2.1722642  2.24442831  2.2667735   2.2734577
## 490   0.78  0.6326799  0.8410809  0.60163418  0.7019047   0.7171708
## 396   4.84  4.0754241  4.1007170  3.98389260  4.0621072   4.0473001
## 137   2.11  2.7045052  2.7040040  2.64823765  2.6810333   2.6679795
## 250   4.24  5.6094995  5.5938475  5.47722480  5.5523496   5.5242074
## 728   4.40  2.3654115  2.3224810  2.37683605  2.3585336   2.3591108
## 567  -0.55  1.0055443  0.8700730  1.07653256  0.9915519   0.9964391
## 291   2.56  2.1649542  2.0975249  2.17868882  2.1495182   2.1494382
## 314   1.89  1.0280913  1.1096438  1.09690595  1.0563701   1.0703954
## 538   2.40  2.5252745  2.3893084  2.45606231  2.5080732   2.4952875
## 233   2.74  1.8892621  1.8695844  1.93005321  1.8767776   1.8784197
## 48    2.80  2.3451066  2.3719617  2.29146253  2.3607217   2.3582021
## 255   2.12  2.6290078  2.5867148  2.64811291  2.6219951   2.6224513
## 118   0.65  1.7276670  1.7770140  1.66650015  1.7223935   1.7153828
## 37    1.17  1.6997905  1.6567359  1.69715654  1.7025710   1.7032533
## 222   2.20  2.1779268  2.3409287  2.09592389  2.2070319   2.2053312
## 658   1.99  1.4408899  1.5814953  1.40128051  1.4795408   1.4847879
## 328   2.17  2.7724363  2.8266098  2.67395092  2.7964137   2.7898571
## 91    1.55  1.2834714  1.3757732  1.35068478  1.3130093   1.3265732
## 584   2.50  3.0723062  2.9570028  3.07891360  3.0327382   3.0239165
## 194   2.06  2.3771641  2.4935166  2.31746643  2.3946980   2.3924004
## 147   2.97  3.1401652  3.0454131  3.16944690  3.0972873   3.0896933
## 663   1.11  1.0083059  0.9350507  1.17407585  1.0226637   1.0448767
## 261   2.41  3.4060294  3.2735141  3.42695716  3.3468716   3.3351619
## 334   3.12  4.0576615  3.9740717  4.06110725  4.0048598   3.9936154
## 296   0.97  1.8877671  1.7552372  1.94022961  1.8853479   1.8906042
## 689   1.44  1.3076068  1.4387667  1.28744123  1.3388091   1.3445277
## 448   2.35  1.7488034  1.7367769  1.74977565  1.7572059   1.7596659
## 743   2.17  1.8904141  1.8380073  1.87161199  1.8891163   1.8870462
## 170   1.94  3.0540137  3.1931932  2.98853310  3.0560106   3.0491865
## 726   1.45  2.3249438  2.3664846  2.28954961  2.3242863   2.3188967
## 336   0.42  1.0868228  1.0067613  1.15384723  1.0842543   1.0914430
## 422   3.69  3.5927189  3.5416192  3.59611033  3.5575563   3.5493344
## 427   0.10  1.7875928  1.7680560  1.79713160  1.7982994   1.8031067
```

```
## 494   1.22  0.4232100  0.4733621  0.49170760  0.4323774   0.4431441
## 316   2.48  1.6041069  1.3388781  1.68638575  1.5510819   1.5489720
## 364   4.45  4.0736992  4.1007170  3.98162546  4.0611438   4.0464100
## 742   1.21  1.3399878  1.2898183  1.38787074  1.3408620   1.3474462
## 100   1.14  2.2500384  2.2292151  2.23207190  2.2187697   2.2102426
## 201   1.12  1.9240934  1.9939012  1.89105983  1.9316255   1.9299147
## 283   0.52  1.1041601  0.7554107  1.25952500  1.0252172   1.0265231
## 71    0.41 -0.2620480 -0.2749377 -0.13902913 -0.2612696  -0.2468588
## 661   2.00  1.7229589  1.6939754  1.80481332  1.7435886   1.7569333
## 551   1.00  0.9178845  0.8658940  0.98926903  0.9304010   0.9425813
## 272   2.26  1.5562968  1.5034869  1.63438513  1.5200924   1.5220349
## 82    3.44  2.6687147  2.5951714  2.71136272  2.6554707   2.6574055
## 648   1.95  1.5528486  1.4315680  1.59856344  1.5116423   1.5088751
## 394   3.88  3.5942151  3.4451815  3.44772350  3.5298636   3.4961009
## 471   2.73  2.2231808  2.0869761  2.29044124  2.2146648   2.2207074
## 210   0.05  0.2980748  0.2342823  0.41780722  0.3385195   0.3615448
## 481   3.37  2.5376438  2.4986473  2.55056915  2.5276160   2.5251034
## 458   0.75  0.7513538  0.8437902  0.77622328  0.7798583   0.7902870
## 177   1.04  1.7395772  1.7789215  1.77593315  1.7497798   1.7556979
## 228   4.85  4.3318143  4.3523447  4.23093325  4.3122025   4.2950888
## 130   2.24  2.0461894  2.1481568  2.03183534  2.0523867   2.0505270
## 721   2.15  2.4999765  2.4759516  2.47520026  2.4913611   2.4878859
## 114   3.19  2.6146452  2.6332742  2.55540167  2.6202064   2.6153196
## 1     0.74  0.5522845  0.6455246  0.59640184  0.5821533   0.5951336
## 457   1.40  0.5522845  0.6455246  0.59640184  0.5821533   0.5951336
## 125   0.60  0.6286674  0.7434905  0.65695425  0.6715033   0.6858361
## 633   0.62  0.3405949  0.4048301  0.39741963  0.3586536   0.3692029
## 269   2.03  3.1310376  3.0204493  3.17325246  3.0412763   3.0260982
## 318   0.62  2.6722589  2.4086077  2.62136301  2.6316268   2.6169549
## 395   3.59  3.0183152  2.9832843  2.95541153  2.9754407   2.9571059
## 704   3.37  2.6346188  2.6062517  2.54086313  2.6450996   2.6347059
## 442   2.75  2.0826680  2.0079493  2.04910596  2.0739022   2.0684669
## 700   1.44  1.9181604  2.0043787  1.92447066  1.9193751   1.9188603
## 675   2.84  2.3848930  2.3619222  2.35569014  2.3810173   2.3778487
## 576   3.60  2.3338935  2.3187625  2.31139478  2.3378709   2.3366230
## 15    2.97  2.6002195  2.5730117  2.59160059  2.5948731   2.5910872
## 276   0.38  2.3955009  2.3938756  2.33148270  2.3812237   2.3712602
## 128   3.02  2.7664229  2.6130425  2.75302050  2.7325812   2.7216109
## 559   3.35  3.0254146  2.9701317  3.05774412  3.0019621   2.9998309
## 770   1.55  1.9019284  1.8380073  1.88339784  1.8974723   1.8947233
## 402   3.18  2.1668838  2.3166438  2.15169113  2.2109836   2.2186878
## 614   1.64  2.4066541  2.4305344  2.37306395  2.3927001   2.3843094
## 473   2.46  1.8806100  1.8192182  1.90517592  1.8724544   1.8750075
## 53    0.81  0.4578719  0.4954659  0.52751994  0.4606234   0.4700166
## 523   0.62  1.5026279  1.4546506  1.53190445  1.5059861   1.5110184
## 205   0.18  0.7382353  0.6767373  0.80431711  0.7510239   0.7609809
```

(b) Compare the predictions using mean squared prediction error. Which model performed the best?

```
set.seed(100)

round(mean((final_pred$full_res - testData$logBCF)^2), 4)
```

```
## [1] 0.5839
```

```
round(mean((final_pred$back_step - testData$logBCF)^2), 4)
```

```
## [1] 0.5742
```

```
round(mean((final_pred$ridge_res - testData$logBCF)^2), 4)
```

```
## [1] 0.5878
```

```
round(mean((final_pred$lasso_res - testData$logBCF)^2), 4)
```

```
## [1] 0.5791
```

```
round(mean((final_pred$elastic_res - testData$logBCF)^2), 4)
```

```
## [1] 0.5783
```

Models performed similarly; the one with the lowest MSE is the model created using backward stepwise regression with BIC.

(c) Provide a table listing each method described in Question 7a and the variables selected by each method (see Lesson 5.8 for an example). Which variables were selected consistently?

	Backward Stepwise	Ridge	Lasso	Elastic Net
nHM	x	x	x	x
piPC09	x	x	x	x
PCD		x	x	x
X2AV		x		
MLOGP	x	x	x	x
ON1V		x	x	x
N.072		x	x	x
B02.C.N.		x	x	x
F04.C.O.	x	x	x	x

nHM, piPC09, MLOGP and F04.C.O are selected consistently.