Harvard_EdX_Project

Colleen Morse

1/4/2022

Introduction

Water pollution affections so many aquatic organisms and can affect food safety for land-based animals including humans. Bioconcntration is the unit by which water pollution is measured in or on an organism. Specifically, the bioconcentration factor (BCF) is the ratio of the concentration of the substance in a specific genus to the exposure concentration, at equilibrium. This study aims to predict a chemical's BCF given 9 molecular descriptors. The data set was obtained from the UCI Machine Learning Repository.

Project completed using: * R version 4.0.3 (2020-10-10) * Platform: x86_64-w64-mingw32/x64 (64-bit) * Running under: Windows 10 x64 (build 19043)

Methods

Load libraries

```
library(CombMSC)
library(boot)
library(leaps)
library(plotmo)
library(MASS)
library(glmnet)
library(tidyverse)
library(ggpubr)
```

Load data

```
library(readr)
data <- read_csv("C:/Users/cmors/Downloads/Grisoni_et_al_2016_EnvInt88.csv",
                 show_col_types = FALSE)

data %>% head(5)
```

```
## # A tibble: 5 x 14
##   CAS      SMILES  Set     nHM piPC09   PCD  X2Av MLOGP  ON1V `N-072` `B02[C-N]`
##   <chr>    <chr>   <chr> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>      <dbl>
## 1 100-02-7 O=[N+]~ Train     0      0  1.49  0.14  1.35  0.72       0          1
```

```
## 2 100-17-4 O=[N+]~ Train     0      0  1.47  0.14  1.7   0.88       0          1
## 3 100-18-5 c1cc(c~ Train     0      0  1.2   0.25  4.14  2.06       0          0
## 4 100-25-4 O=[N+]~ Train     0      0  1.69  0.13  1.89  0.79       0          1
## 5 100-40-3 C=CC1C~ Train     0      0  0.52  0.25  2.65  1.31       0          0
## # ... with 3 more variables: F04[C-O] <dbl>, Class <dbl>, logBCF <dbl>
```

Exploratory Data Analysis

```
data %>% dim()
```

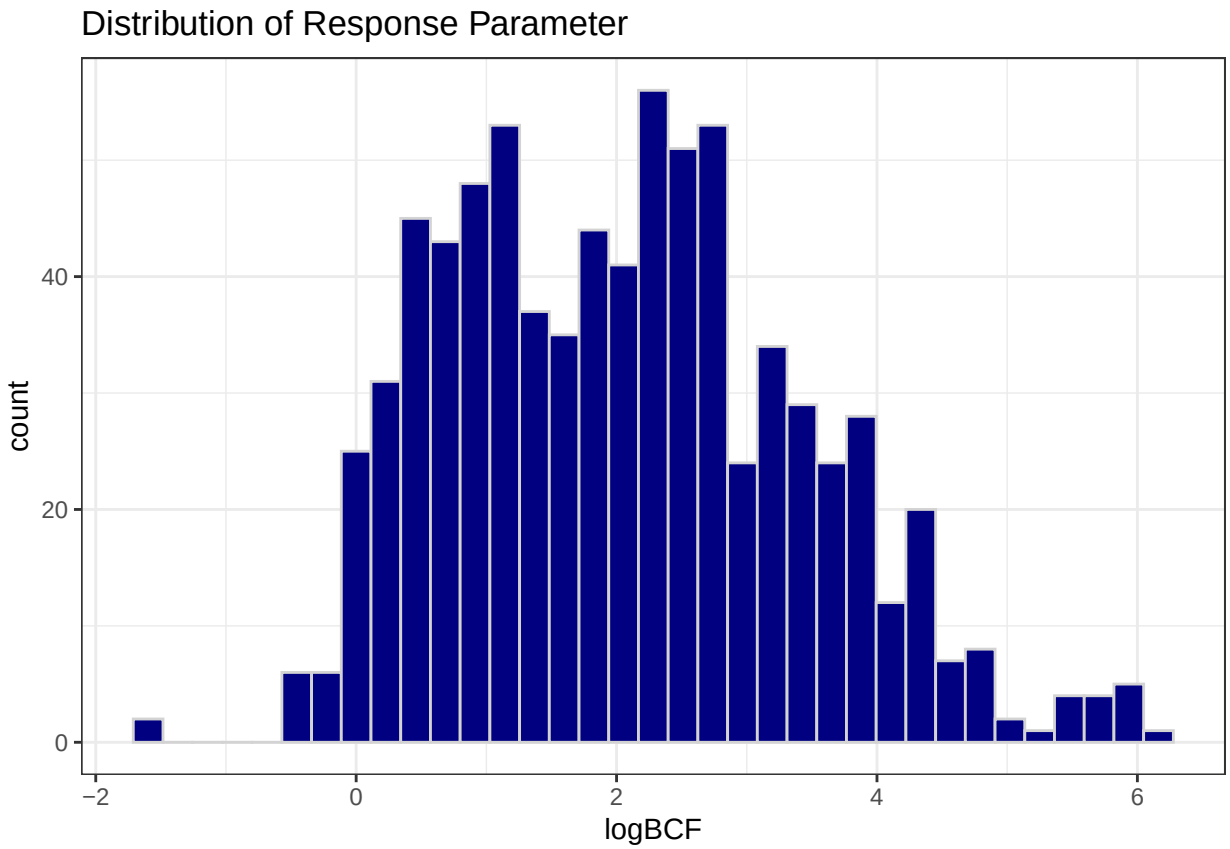
```
## [1] 779  14
```

The data contains information for 779 molecular compounds, identified by the id number and notation given in the first and second columns, CAS and SMILES. The third column, Set, identifies the compound as either the "training" set (75%) or the "test" set (25%). This designation was done randomly while keeping distributions of qualities similar in each group. Columns 4 through 12 are various molecular descriptors. - *nHM*: number of heavy atoms (integer) - *piPC09*: molecular multiple path count (numeric) - *PCD*: difference between multiple path cound and path count (numeric) - *X2Av*: average valence connectivity (numeric) - *MLOGP*: Moriguchi octanol-water partition coefficient (numeric) - *ON1V*: overall modified Zagreb index by valence vertex degrees (numeric) - *N-072*: Frequency of RCO-N< / >N-X=X fragments (integer) - *B02[C-N]*: Presence / Absence of C-N atom pairs (binary) - *F04[C-O]*: Frequency of C-O atom pairs (integer) Column 13 refers to how the molecule is digested (1 = is mainly stored within lipid tissues, 2 = has additional storage sites (e.g. proteins), or 3 = is metabolized/eliminated). Column 14 is the Bioconcentration Factor (BCF) in log units.

Both columns 13 and 14 are response variables for this data set. Column 14 is the column of interest for this study.

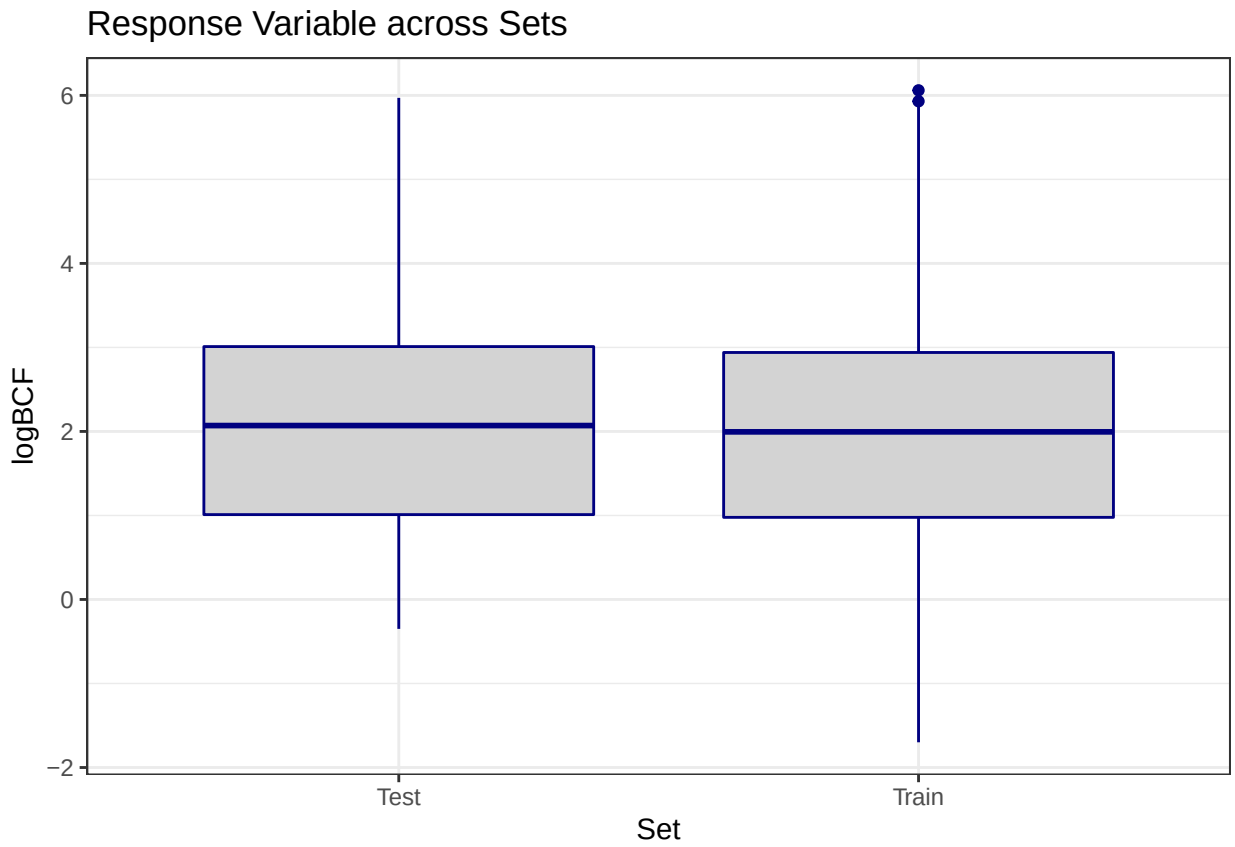
```
# remove ID columns and classification response column
data <- data %>% dplyr::select(-c(CAS, SMILES, Class))
```

```
data %>%
  ggplot(aes(logBCF)) +
  geom_histogram(bins = 35, fill = 'navyblue', color = 'lightgrey') +
  theme_bw() +
  labs(title = 'Distribution of Response Parameter')
```



The response variable appears it may have a bi-modal distribution.

```
data %>%
  ggplot(aes(Set, logBCF)) +
  geom_boxplot(fill = 'lightgrey', color = 'navyblue') +
  theme_bw() +
  labs(title = 'Response Variable across Sets')
```



The response variable appears to be well balanced.

```
g1 <- data %>% ggplot(aes(nHM, logBCF)) +
  geom_point(color = 'navyblue', alpha = 0.5) +
  theme_bw() +
  labs(title = 'nHM and logBCF')

g2 <- data %>% ggplot(aes(piPC09, logBCF)) +
  geom_point(color = 'navyblue', alpha = 0.5) +
  theme_bw() +
  labs(title = 'piPC09 and logBCF')

g3 <- data %>% ggplot(aes(PCD, logBCF)) +
  geom_point(color = 'navyblue', alpha = 0.5) +
  theme_bw() +
  labs(title = 'PCD and logBCF')

g4 <- data %>% ggplot(aes(X2Av, logBCF)) +
  geom_point(color = 'navyblue', alpha = 0.5) +
  theme_bw() +
  labs(title = 'X2Av and logBCF')

g5 <- data %>% ggplot(aes(MLOGP, logBCF)) +
  geom_point(color = 'navyblue', alpha = 0.5) +
  theme_bw() +
  labs(title = 'MLOGP and logBCF')
```

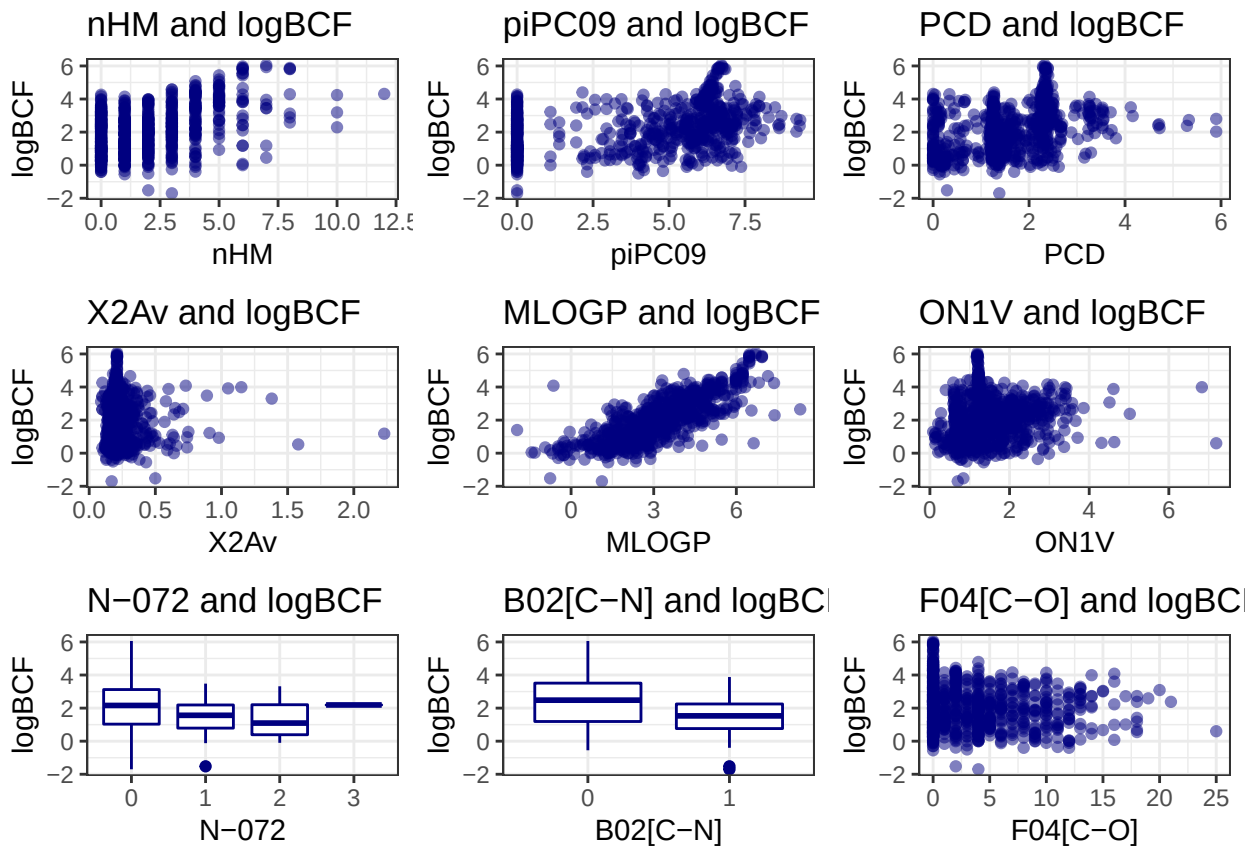
```
g6 <- data %>% ggplot(aes(ON1V, logBCF)) +
  geom_point(color = 'navyblue', alpha = 0.5) +
  theme_bw() +
  labs(title = 'ON1V and logBCF')

g7 <- data %>% ggplot(aes(as.factor(`N-072`), logBCF)) +
  geom_boxplot(color = 'navyblue') +
  theme_bw() +
  labs(title = 'N-072 and logBCF',
       x = 'N-072')

g8 <- data %>% ggplot(aes(as.factor(`B02[C-N]`), logBCF)) +
  geom_boxplot(color = 'navyblue') +
  theme_bw() +
  labs(title = 'B02[C-N] and logBCF',
       x = 'B02[C-N]')

g9 <- data %>% ggplot(aes(`F04[C-O]`, logBCF)) +
  geom_point(color = 'navyblue', alpha = 0.5) +
  theme_bw() +
  labs(title = 'F04[C-O] and logBCF')

ggarrange(g1, g2, g3, g4, g5, g6, g7, g8, g9,
          ncol = 3, nrow = 3)
```



MLOGP appears to have a near-linear relationship with logBCF.

Split into Train / Test sets via the "Set" parameter

```
table(data$Set)
```

```
##
##  Test Train
##   195   584
```

```
data_train <- data %>% filter(Set == 'Train') %>% dplyr::select(-Set)
data_test <- data %>% filter(Set == 'Test') %>% dplyr::select(-Set)
```

```
head(data_train, 5)
```

```
## # A tibble: 5 x 10
##     nHM piPC09   PCD  X2Av MLOGP  ON1V `N-072` `B02[C-N]` `F04[C-O]` logBCF
##   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>      <dbl>      <dbl>  <dbl>
## 1     0      0  1.49  0.14  1.35  0.72       0          1          5   0.74
## 2     0      0  1.47  0.14  1.7   0.88       0          1          5   0.93
## 3     0      0  1.2   0.25  4.14  2.06       0          0          0   3.24
## 4     0      0  1.69  0.13  1.89  0.79       0          1          8  -0.4
## 5     0      0  0.52  0.25  2.65  1.31       0          0          0   2.24
```

Model1: standard linear regression

```
model1 <- lm(logBCF ~ ., data = data_train)

summary(model1)
```

```
##
## Call:
## lm(formula = logBCF ~ ., data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0337 -0.5016  0.0937  0.4970  4.1320
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.083046   0.137242   0.605 0.545346
## nHM          0.135033   0.022762   5.932 5.17e-09 ***
## piPC09       0.043511   0.020992   2.073 0.038641 *
## PCD          0.019941   0.059874   0.333 0.739219
## X2Av        -0.112300   0.262555  -0.428 0.669014
## MLOGP        0.500488   0.033950  14.742  < 2e-16 ***
## ON1V         0.110698   0.067384   1.643 0.100977
## `N-072`     -0.119943   0.075092  -1.597 0.110751
## `B02[C-N]`  -0.114887   0.081170  -1.415 0.157497
## `F04[C-O]`  -0.033176   0.009478  -3.500 0.000501 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7832 on 574 degrees of freedom
## Multiple R-squared:  0.6616, Adjusted R-squared:  0.6562
## F-statistic: 124.7 on 9 and 574 DF,  p-value: < 2.2e-16
```

```
summary(model1)$coefficients[summary(model1)$coefficients[,4] <= 0.05, ]
```

```
##               Estimate  Std. Error   t value     Pr(>|t|)
## nHM         0.13503310 0.022762333  5.932305 5.165683e-09
## piPC09      0.04351099 0.020991931  2.072748 3.864142e-02
## MLOGP       0.50048840 0.033950474 14.741720 6.051546e-42
## `F04[C-O]` -0.03317586 0.009477694 -3.500414 5.006034e-04
```

Coefficients for nHM, piPC09, MLOGP, and F04[C-O] are significant at a 95% confidence level.

```
# Cross validation scores:
set.seed(100)

n <- nrow(data_train)

model1_glm <- glm(logBCF ~ ., data = data_train)

data.frame(ten_fold = cv.glm(data_train, model1_glm, K = 10)$delta[1],
           loocv = cv.glm(data_train, model1_glm, K=n)$delta[1])
```

```
##    ten_fold     loocv
## 1 0.6384414 0.6337806
```

10-fold CV MSE: 0.6330498 Leave-One-Out CV MSE: 0.6337806

```
# Mallow's CP, AIC, and BIC scores:
set.seed(100)

data.frame(Mallows_CP = Cp(model1, S2 = summary(model1)$sigma^2),
           AIC = AIC(model1, k = 2),
           BIC = AIC(model1, k = log(nrow(data_train))))  # BIC: k = log(n)
```

```
##   Mallows_CP      AIC      BIC
## 1         10 1383.849 1431.918
```

Mallow's CP: 10 (to be used for model2) AIC: 1383.849 (to be used for model3a) BIC: 1431.918 (to be used for model3b)

```
# Compare all possible models using Mallow's CP

cat('There are', 2^(ncol(data)-1), 'possible models.')
```

```
## There are 1024 possible models.
```

```r
set.seed(100)

out <- leaps(data_train %>% dplyr::select(-logBCF),
             data_train$logBCF,
             method = 'Cp',
             nbest = 1)

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

```
##   1 2 3 4 5 6 7 8 9
## 1 0 0 0 0 1 0 0 0 0 58.970660
## 2 1 0 0 0 1 0 0 0 0 20.613835
## 3 1 1 0 0 1 0 0 0 0 16.923788
## 4 1 1 0 0 1 0 0 0 1  7.705310
## 5 1 1 0 0 1 0 1 0 1  6.542419
## 6 1 1 0 0 1 1 1 0 1  6.037255
## 7 1 1 0 0 1 1 1 1 1  6.342960
## 8 1 1 0 1 1 1 1 1 1  8.110922
## 9 1 1 1 1 1 1 1 1 1 10.000000
```

The above grid outlines the Mallow's CP score associated with the best model associated with each possible total number of parameters. For example, the first row shows the best model that uses only one predictor and is the model that used the 5th parameter, or MLOGP. The best model that used only two parameters used both the 1st and 5th parameter: nHM and MLOGP. The row that is associated with the lowest Mallow's CP is the 6th row where 6 parameters were used; namely: nHM, piPC09, MLOGP, ON1V, N-072, and F04[C-O].

Model2: use only the parameters identified by Mallow's CP

```r
set.seed(100)

model2 <- lm(logBCF ~ nHM + piPC09 + MLOGP + ON1V + `N-072` + `F04[C-O]`,
             data = data_train)

summary(model2)
```

```
##
## Call:
## lm(formula = logBCF ~ nHM + piPC09 + MLOGP + ON1V + `N-072` +
##     `F04[C-O]`, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9650 -0.4926  0.0709  0.5153  4.1887
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.008318   0.089936   0.092 0.926344
## nHM          0.131750   0.019751   6.671 5.98e-11 ***
## piPC09       0.047722   0.014178   3.366 0.000814 ***
## MLOGP        0.518510   0.029932  17.323  < 2e-16 ***
```

```
## ON1V         0.089186   0.056301   1.584 0.113720
## `N-072`     -0.148490   0.071483  -2.077 0.038218 *
## `F04[C-O]`  -0.032499   0.009112  -3.567 0.000392 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7826 on 577 degrees of freedom
## Multiple R-squared:  0.6604, Adjusted R-squared:  0.6568
## F-statistic:   187 on 6 and 577 DF,  p-value: < 2.2e-16
```

Model3a: create model with forward stepwise regression using AIC

```
set.seed(100)

# the minimum possible model is that of only an intercept
min_model <- lm(logBCF ~ 1, data = data_train)

model3a <- step(min_model,
                scope = list(lower = min_model,
                             upper = model1),
                direction = 'forward',
                k = 2,
                trace = FALSE)

summary(model3a)
```

```
##
## Call:
## lm(formula = logBCF ~ MLOGP + nHM + piPC09 + `F04[C-O]` + `N-072` +
##     ON1V, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9650 -0.4926  0.0709  0.5153  4.1887
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.008318   0.089936   0.092 0.926344
## MLOGP        0.518510   0.029932  17.323  < 2e-16 ***
## nHM          0.131750   0.019751   6.671 5.98e-11 ***
## piPC09       0.047722   0.014178   3.366 0.000814 ***
## `F04[C-O]`  -0.032499   0.009112  -3.567 0.000392 ***
## `N-072`     -0.148490   0.071483  -2.077 0.038218 *
## ON1V         0.089186   0.056301   1.584 0.113720
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7826 on 577 degrees of freedom
## Multiple R-squared:  0.6604, Adjusted R-squared:  0.6568
## F-statistic:   187 on 6 and 577 DF,  p-value: < 2.2e-16
```

6 parameters were selected by forward stepwise regression: MLOP, nHM, piPC09, F04[C-O], N-072, and ON1V.

Model3b: create model with backward stepwise regression using BIC

```
set.seed(100)

model3b <- step(model1,   # start with full model and work backward
                scope = list(lower = min_model,
                             upper = model1),
                direction = 'backward',
                k = log(nrow(data_train)),
                trace = FALSE)

summary(model3b)
```

```
##
## Call:
## lm(formula = logBCF ~ nHM + piPC09 + MLOGP + `F04[C-O]`, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8932 -0.4936  0.0645  0.5017  4.3582
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02254    0.08027   0.281 0.778969
## nHM          0.11673    0.01807   6.459 2.24e-10 ***
## piPC09       0.04570    0.01353   3.377 0.000781 ***
## MLOGP        0.55041    0.02671  20.608  < 2e-16 ***
## `F04[C-O]`  -0.02596    0.00777  -3.342 0.000887 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7851 on 579 degrees of freedom
## Multiple R-squared:  0.657,  Adjusted R-squared:  0.6546
## F-statistic: 277.3 on 4 and 579 DF,  p-value: < 2.2e-16
```

Only 4 parameters remain after performing backward stepwise regression; these are the same 4 parameters as were found to be significant at a 95% confidence interval in model1.

Model4: create model from LASSO regression

```
predictors <- data_train[,1:9]
response <- data_train$logBCF
```

```
set.seed(100)

lasso_cv <- cv.glmnet(as.matrix(predictors),
```

```
                        response,
                        nfolds = 10,
                        alpha = 1)     # alpha = 1 for LASSO regression

lasso_cv
```

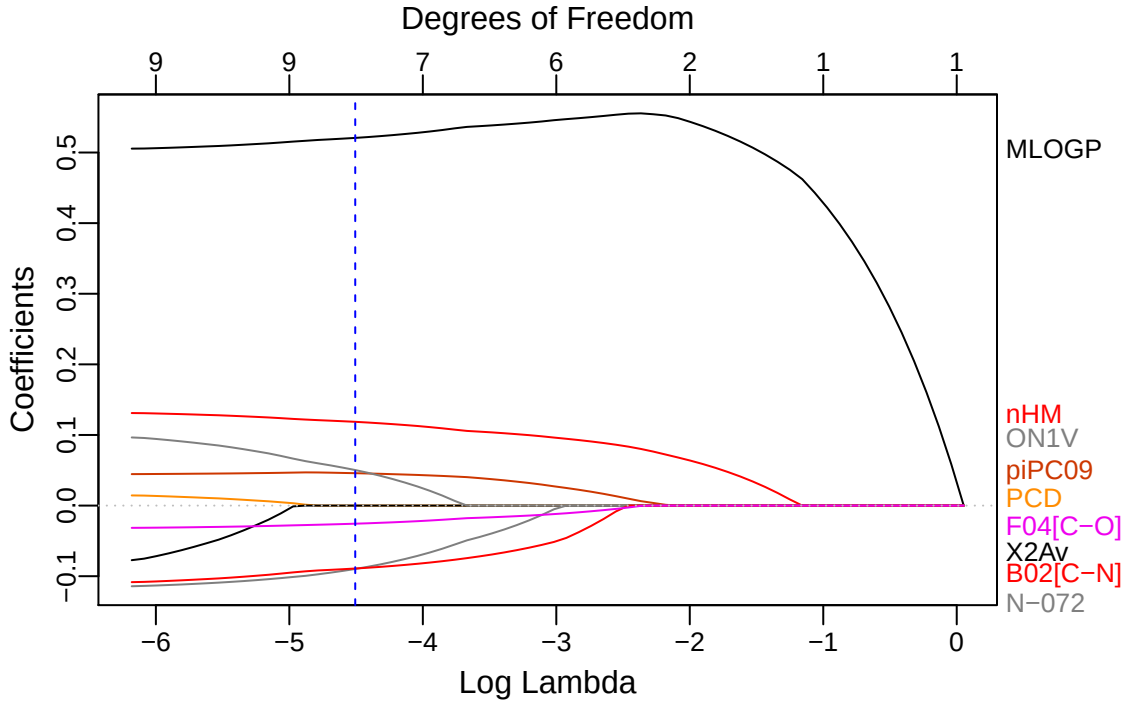
```
##
## Call:  cv.glmnet(x = as.matrix(predictors), y = response, nfolds = 10,      alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.01104    50  0.6245 0.05466       7
## 1se 0.16392    21  0.6722 0.05707       2
```

The lambda value that minimizes error is 0.01212, which corresponds to an MSE of 0.6337.

```
lasso_model <- glmnet(as.matrix(predictors),
                        response,
                        alpha = 1,
                        nlambda = 100)
```

```
plot_glmnet(lasso_model, xvar = 'lambda', label = TRUE)

abline(v = log(lasso_cv$lambda.min),
       col = 'blue', lty = 2)
```



MLOGP was the first parameter selected followed by nHM, then piPC09, F04[C-O], and so on.

```
coef(lasso_model, s = lasso_cv$lambda.min)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  0.09043968
## nHM          0.11859806
## piPC09       0.04588928
## PCD          .
## X2Av         .
## MLOGP        0.52066639
## ON1V         0.05035471
## N-072       -0.08976321
## B02[C-N]    -0.08910951
## F04[C-O]    -0.02565006
```

7 coefficients were included in the LASSO model at the optimal lambda value.

```
# create linear regression model from the 7 predictors

model4 <- lm(logBCF ~ nHM + piPC09 + MLOGP + ON1V +
               `N-072` + `B02[C-N]` + `F04[C-O]`,
             data = data_train)

summary(model4)
```

```
##
## Call:
## lm(formula = logBCF ~ nHM + piPC09 + MLOGP + ON1V + `N-072` +
##     `B02[C-N]` + `F04[C-O]`, data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9993 -0.5125  0.0905  0.5010  4.1184
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.070727   0.101837   0.695 0.487645
## nHM          0.128632   0.019883   6.469  2.1e-10 ***
## piPC09       0.050199   0.014296   3.511 0.000481 ***
## MLOGP        0.508381   0.030906  16.449  < 2e-16 ***
## ON1V         0.089809   0.056269   1.596 0.111024
## `N-072`     -0.121050   0.074477  -1.625 0.104639
## `B02[C-N]`  -0.100382   0.077008  -1.304 0.192915
## `F04[C-O]`  -0.031720   0.009126  -3.476 0.000548 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7821 on 576 degrees of freedom
## Multiple R-squared:  0.6613, Adjusted R-squared:  0.6572
## F-statistic: 160.7 on 7 and 576 DF,  p-value: < 2.2e-16
```

Model5: create model from Elastic Net

```
set.seed(100)

elastic_weights <- NULL

for (i in seq(0.1, 0.9, 0.1)) {
  elastic_cv <- cv.glmnet(as.matrix(predictors),
                          response,
                          nfolds = 10,
                          alpha = i)

  elastic_weights <- rbind(elastic_weights, c(weight = i,
                                              lambda = elastic_cv$lambda.min))
}

elastic_weights[elastic_weights[,2] == min(elastic_weights[,2]), ]
```

```
##     weight     lambda
## 0.80000000 0.01257263
```

An alpha of 0.8 is associated with the lowest 10-fold cross validation score for elastic net.

```
set.seed(100)

elastic_cv <- cv.glmnet(as.matrix(predictors),
                        response,
                        nfolds = 10,
                        alpha = 0.8)

elastic_cv
```

```
##
## Call:  cv.glmnet(x = as.matrix(predictors), y = response, nfolds = 10,      alpha = 0.8)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure      SE Nonzero
## min 0.0138    50  0.6245 0.05459       7
## 1se 0.1867    22  0.6724 0.05627       2
```

```
elastic_model <- glmnet(as.matrix(predictors),
                        response,
                        alpha = 0.8,
                        nlambda = 100)

coef(elastic_model, s = elastic_cv$lambda.min)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  0.09474703
```

```
## nHM          0.11884728
## piPC09       0.04626270
## PCD          .
## X2Av         .
## MLOGP        0.51874927
## ON1V         0.05116924
## N-072       -0.09058518
## B02[C-N]    -0.09061287
## F04[C-O]    -0.02574529
```

The elastic net model chose the same 7 predictors as the LASSO regression model. Model5 therefore would be equivalent to model4.

Results

Compare the models' adjusted R^2, Mallow's CP, AIC, and BIC scores

```
rbind(model1 = c(Adj_R_2 = summary(model1)$adj.r.squared,
                 Mallows_CP = Cp(model1, S2 = summary(model1)$sigma^2),
                 AIC = AIC(model1, k=2),
                 BIC = AIC(model1, k=log(nrow(data_train)))),
      model2 = c(Adj_R_2 = summary(model2)$adj.r.squared,
                 Mallows_CP = Cp(model2, S2 = summary(model2)$sigma^2),
                 AIC = AIC(model2, k=2),
                 BIC = AIC(model2, k=log(nrow(data_train)))),
      model3a = c(Adj_R_2 = summary(model3a)$adj.r.squared,
                 Mallows_CP = Cp(model3a, S2 = summary(model3a)$sigma^2),
                 AIC = AIC(model3a, k=2),
                 BIC = AIC(model3a, k=log(nrow(data_train)))),
      model3b = c(Adj_R_2 = summary(model3b)$adj.r.squared,
                 Mallows_CP = Cp(model3b, S2 = summary(model3b)$sigma^2),
                 AIC = AIC(model3b, k=2),
                 BIC = AIC(model3b, k=log(nrow(data_train)))),
      model4 = c(Adj_R_2 = summary(model4)$adj.r.squared,
                 Mallows_CP = Cp(model4, S2 = summary(model4)$sigma^2),
                 AIC = AIC(model4, k=2),
                 BIC = AIC(model4, k=log(nrow(data_train))))
      )
```

```
##           Adj_R_2 Mallows_CP      AIC      BIC
## model1  0.6562449         10 1383.849 1431.918
## model2  0.6568185          7 1379.918 1414.878
## model3a 0.6568185          7 1379.918 1414.878
## model3b 0.6546388          5 1381.637 1407.856
## model4  0.6572338          8 1380.198 1419.527
```

While model3b has the lowest R^2 value, it shows the best score for Mallow's CP and prediction risk (BIC) and has fairly close AIC and R^2 scores compared to the other models. BIC is the preferred metric for prediction risk due to its higher penalty associated with more complex models.

Compare the models' prediction accuracies

Training Errors (MSE):

```
set.seed(100)

# Full model, linear regression
pred_full <- predict(model1, data_train,
                     interval = 'prediction')[,1]

# Mallow's CP, linear regression
pred_cp <- predict(model2, data_train,
                   interval = 'prediction')[,1]

# forward stepwise regression with AIC, linear regression
pred_forward <- predict(model3a, data_train,
                        interval = 'prediction')[,1]

# backward stepwise regression with BIC, linear regression
pred_backward <- predict(model3b, data_train,
                         interval = 'prediction')[,1]

# LASSO regression
pred_lasso <- predict.glmnet(lasso_model, as.matrix(data_train[,1:9]),
                             s = lasso_cv$lambda.min,
                             interval = 'prediction')[,1]

# Elastic Net regression
pred_elastic <- predict.glmnet(elastic_model, as.matrix(data_train[,1:9]),
                               s = elastic_cv$lambda.min,
                               interval = 'prediction')[,1]

train_predictions <- cbind(true = data_train$logBCF,
                           pred_full, pred_cp, pred_forward,
                           pred_backward, pred_lasso, pred_elastic)
```

```
mse <- function(a, b){
  return(mean((a-b)^2))
}

train_mse <- NULL

for (i in 1:(ncol(train_predictions[,-1]))) {
  model_name <- colnames(train_predictions)[i+1]
  model_mse <- mse(train_predictions[,1], train_predictions[,i+1])
  train_mse <- rbind(train_mse, c(model_name, model_mse))
}

train_mse
```

```
##      [,1]           [,2]
## [1,] "pred_full"    "0.602936591036922"
```

```
## [2,] "pred_cp"       "0.60507654888071"
## [3,] "pred_forward"  "0.60507654888071"
## [4,] "pred_backward" "0.611030342181451"
## [5,] "pred_lasso"    "0.604342870224707"
## [6,] "pred_elastic"  "0.604328989833186"
```

The full linear regression model has the lowest training error of all models.

Testing Errors (MSE):

```
set.seed(100)

# Full model, linear regression
pred_full <- predict(model1, data_test,
                     interval = 'prediction')[,1]

# Mallow's CP, linear regression
pred_cp <- predict(model2, data_test,
                   interval = 'prediction')[,1]

# forward stepwise regression with AIC, linear regression
pred_forward <- predict(model3a, data_test,
                        interval = 'prediction')[,1]

# backward stepwise regression with BIC, linear regression
pred_backward <- predict(model3b, data_test,
                         interval = 'prediction')[,1]

# LASSO regression
pred_lasso <- predict.glmnet(lasso_model, as.matrix(data_test[,1:9]),
                             s = lasso_cv$lambda.min,
                             interval = 'prediction')[,1]

# Elastic Net regression
pred_elastic <- predict.glmnet(elastic_model, as.matrix(data_test[,1:9]),
                               s = elastic_cv$lambda.min,
                               interval = 'prediction')[,1]

test_predictions <- cbind(true = data_test$logBCF,
                          pred_full, pred_cp, pred_forward,
                          pred_backward, pred_lasso, pred_elastic)
```

```
test_mse <- NULL

for (i in 1:(ncol(test_predictions[,-1]))) {
  model_name <- colnames(test_predictions)[i+1]
  model_mse <- mse(test_predictions[,1], test_predictions[,i+1])
  test_mse <- rbind(test_mse, c(model_name, model_mse))
}

test_mse
```

```
##      [,1]            [,2]
## [1,] "pred_full"     "0.654034732048738"
## [2,] "pred_cp"       "0.658716297062111"
## [3,] "pred_forward"  "0.658716297062111"
## [4,] "pred_backward" "0.653429247686185"
## [5,] "pred_lasso"    "0.64956118168742"
## [6,] "pred_elastic"  "0.649847581350239"
```

Unlike training errors, testing errors show the LASSO model performed best.

```
test_mse[test_mse[,2] == min(test_mse[,2]), ]
```

```
## [1] "pred_lasso"       "0.64956118168742"
```

Compare parameters chosen by each model

	Full Linear	CP Linear	Forward Stepwise	Backward Stepwise	LASSO	Elastic Net
nHM	yes	yes	yes	yes	yes	yes
piPC09	yes	yes	yes	yes	yes	yes
PCD	yes	-	-	-	-	-
X2AV	yes	-	-	-	-	-
MLOGP	yes	yes	yes	yes	yes	yes
ON1V	yes	yes	yes	-	yes	yes
N-072	yes	yes	yes	-	yes	yes
B02[C-N]	yes	-	-	-	yes	yes
F04[C-O]	yes	yes	yes	yes	yes	yes

The parameters: nHM, piPC09, MLOGP, and FO4[C-O] were included in every model tested. The model that used backward stepwise regression with BIC prior to linear regression was the smallest model. This model had the 3rd lowest testing error of all models tested, behind LASSO and Elastic Net models. LASSO and Elastic Net models performed very similarly using the testing error MSE metric, however the LASSO model performed slightly better.

Conclusion

The model produced from LASSO regression was found to have the lowest testing error in this study, using 7 of the 9 available predictors: nHM, piPC09, MLOGP, ON1V, N-072, B02[C-N], and F04[C-O]. The model produced from Elastic Net regression performed very similarly and used the same 7 parameters. The model formed from backward stepwise regression with BIC prior to linear regression had a similar testing error using only 4 parameters: nHM, piPC09, MLOGP, and FO4[C-O]. If data collection for parameters is costly, one could use the backward stepwise regression model without sacrificing much in the way of error.

References

1. Grisoni, F., Consonni, V., Vighi, M., Villa, S., & Todeschini, R. (2016). Investigating the mechanisms of bioconcentration through QSAR classification trees. Environment international, 88, 198–205. https://doi.org/10.1016/j.envint.2015.12.024

2. Grisoni, F., Consonni, V., Villa, S., Vighi, M., & Todeschini, R. (2015). QSAR models for bioconcentration: is the increase in the complexity justified by more accurate predictions?. Chemosphere, 127, 171–179. https://doi.org/10.1016/j.chemosphere.2015.01.047

3. Data sourced from: https://archive.ics.uci.edu/ml/datasets/QSAR+Bioconcentration+classes+dataset on 01/04/2021.