STAT 534: Homework

Erick Calderon-Morales

Fall 2021

```
library(tidyverse)
library(tidymodels)
library(MASS)
library(janitor)
library(GGally)
library(pls)
library(leaps)
library(glmnet)
```

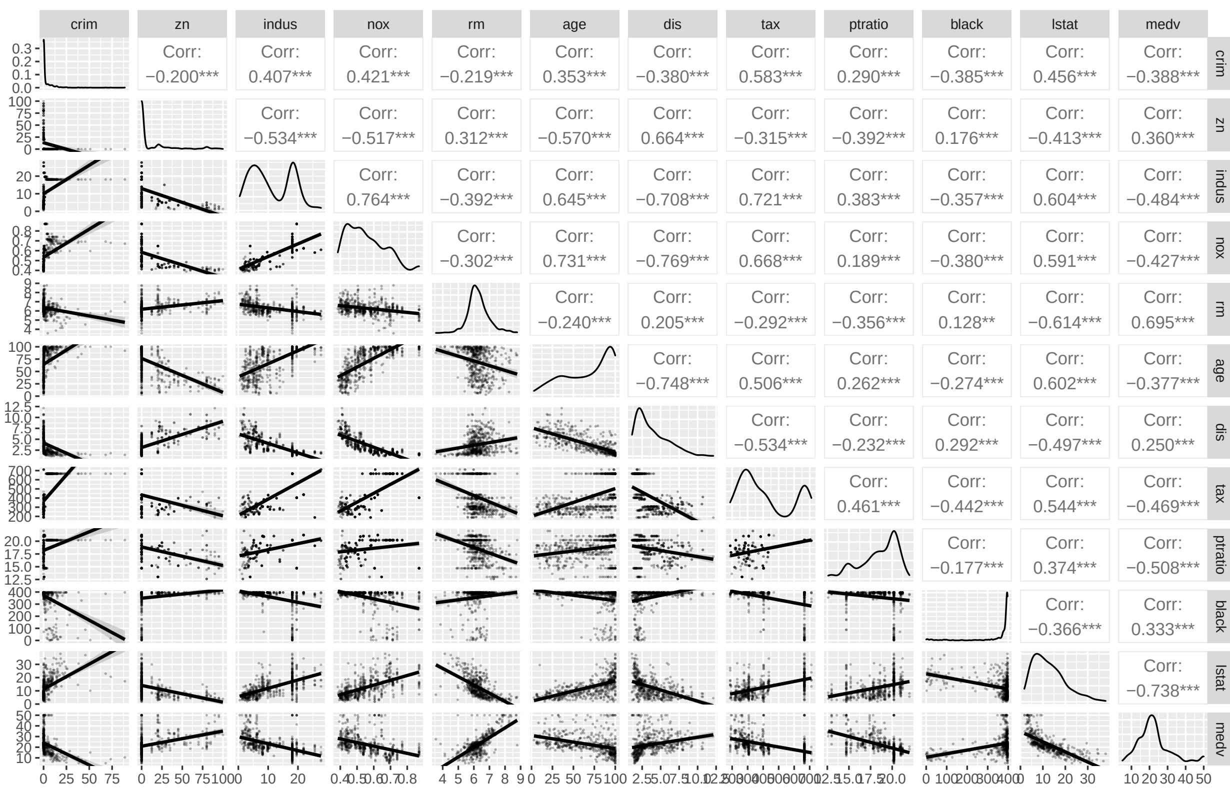
1. Using the Boston dataset from the MASS package, the goal is to predict the crime rate by the other variables.

```
# load data
data(Boston)

Boston <- Boston %>%
  # Transform to factor
  mutate(across(where(is.integer), as.factor)) %>%
  clean_names() %>%
  drop_na()

str(Boston)
```

```
'data.frame':   506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : Factor w/ 9 levels "1","2","3","4",..: 1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

(a) Create bivariate plots to explore relations between variables. Comment on your observations. (Hint: you may use ggpair() and remember to factorize any categorical variables.)

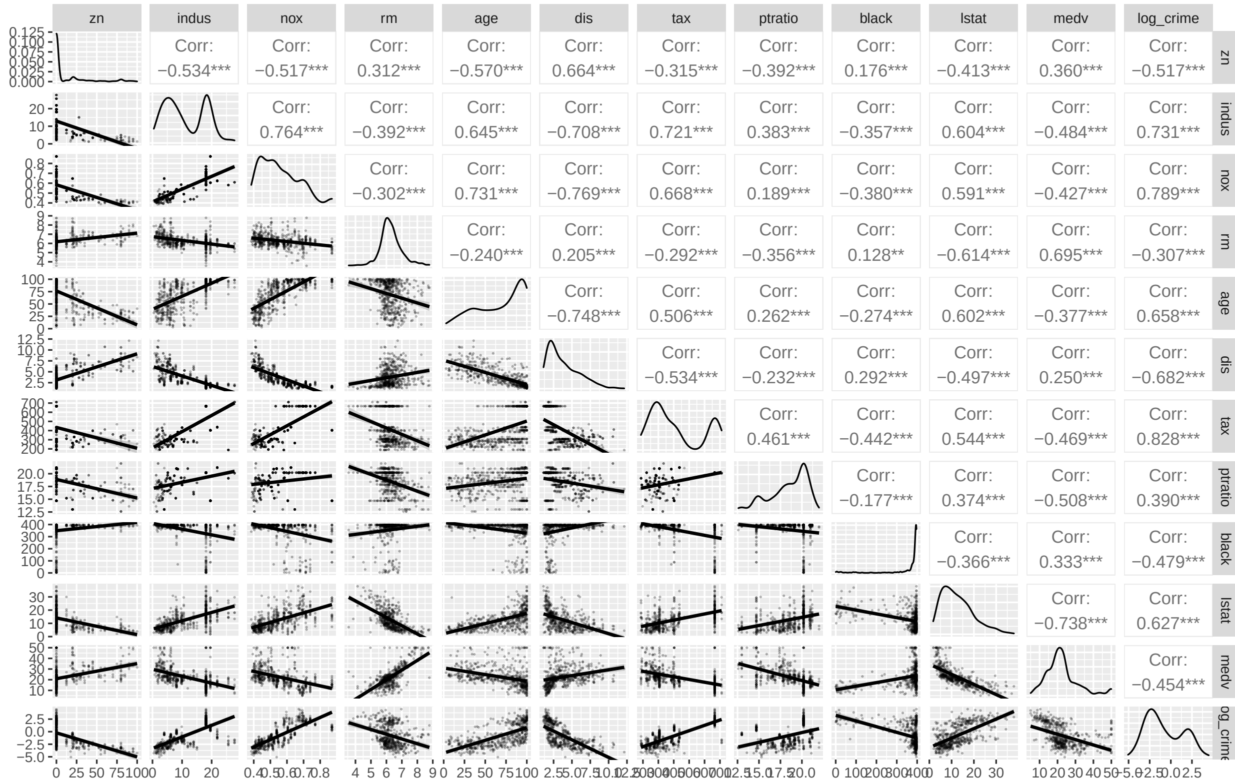
```
Boston %>%
  select_if(is.numeric) %>%
  ggpairs(lower = list(continuous = wrap("smooth", alpha = 0.3, size = 0.1)))
```



For the variables explored, all have a significant correlation with crime, being nox, indus, tax and lstat the ones with the highest positive correlation.

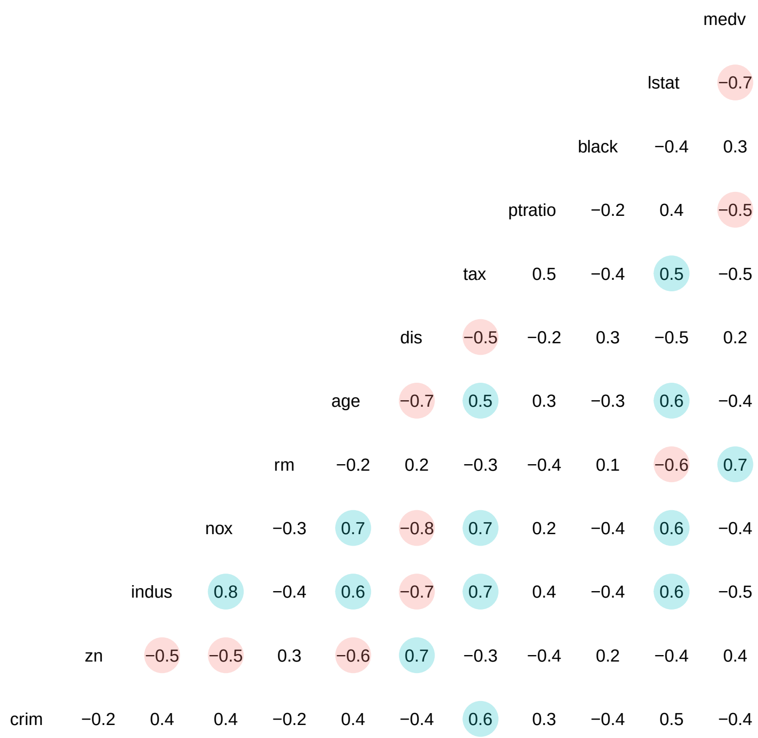
(b) Log-transform the crime rate and repeat part (a).

```
Boston %>%
  mutate(across(where(is.integer), as.factor)) %>%
  mutate(log_crime = log(crim)) %>%
  dplyr::select(-crim) %>%
  clean_names() %>%
  select_if(is.numeric) %>%
  ggpairs(lower = list(continuous = wrap("smooth", alpha = 0.3, size = 0.1)))
```



(c) Create a correlation matrix of all the continuous variables and make comments. (Hint: you may use ggcorr())

```r
# Nice visualization of correlations
Boston %>%
  mutate(across(where(is.integer), as.factor)) %>%
  clean_names() %>%
  select_if(is.numeric) %>%
  ggcorr(geom = "blank", label = TRUE, hjust = 0.75) +
    geom_point(size = 10, aes(color = coefficient > 0, alpha = abs(coefficient) > 0.5)) +
    scale_alpha_manual(values = c("TRUE" = 0.25, "FALSE" = 0)) +
    guides(color = FALSE, alpha = FALSE)
```



(d) Split the data into training and testing subsets.

```
set.seed (666)

# Get y and model matrix
x_boston <- model.matrix(crim ~ . ,Boston)[, -1]
y_boston <- Boston$crim

# Create the training data 80% training
train_boston <- sample(1:nrow(x_boston), 0.80*nrow(x_boston))

# Create the test data 25%
test_boston <- (-train_boston)

# Check percentages
# Train
(nrow(x_boston[train_boston,])/nrow(Boston))*100
```

```
[1] 79.8419
```

```
# Test
(nrow(x_boston[test_boston,])/nrow(Boston))*100
```

```
[1] 20.1581
```

```
# Response variable from train and test datasets
y_test <- y_boston[-train_boston]
y_train <- y_boston[train_boston]
```

(e) Build the following models using the training set:

- **Multiple linear regression**

```
multiple_lm <- lm(crim ~ ., data = Boston[train_boston,])
```

- **Principal Components Regression (indicate how many principal components are selected)**

```
pcr_fit_crime <- pcr(crim ~ .,
                  subset = train_boston,
                  scale = TRUE,
                  validation = "CV",
                  data = Boston )

summary(pcr_fit_crime)
```

```
Data:   X dimension: 404 20
    Y dimension: 404 1
Fit method: svdpc
Number of components considered: 20

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           8.805    7.320    7.328    6.933    6.846    6.843    6.842
adjCV        8.805    7.316    7.325    6.926    6.839    6.836    6.836
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       6.834    6.824    6.795     6.791     6.806     6.808     6.810
adjCV    6.827    6.817    6.789     6.784     6.798     6.800     6.802
       14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
CV        6.779     6.697     6.684     6.677     6.670     6.660     6.624
adjCV     6.811     6.687     6.675     6.668     6.657     6.647     6.611

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X       32.16    42.38    50.00    56.57    62.41    67.92    73.29    78.12
crim    31.63    31.77    39.72    41.18    41.28    41.28    41.42    41.51
      9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X       82.68     86.67     89.94     92.56     94.48     95.75     97.00
crim    42.22     42.45     42.54     42.59     42.72     42.74     44.92
      16 comps  17 comps  18 comps  19 comps  20 comps
X        97.96     98.84     99.46     99.84    100.00
crim     45.12     45.23     46.56     46.78     47.41
```

In this case the lowest CV value (6.624) was obtained with 19 principal components

- **Partial Least Squares (indicate how many directions are selected)**

```
set.seed (666)
pls_fit_crime <- plsr(crim ~ .,
                      subset = train_boston,
                      scale = TRUE,
                      validation = "CV",
                      data = Boston)
summary(pls_fit_crime)
```

```
Data:   X dimension: 404 20
    Y dimension: 404 1
```

```
Fit method: kernelpls
Number of components considered: 20

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           8.805    7.095    6.752    6.759    6.721    6.703    6.680
adjCV        8.805    7.091    6.745    6.742    6.705    6.686    6.665
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       6.656    6.667    6.665     6.658     6.655     6.652     6.651
adjCV    6.642    6.651    6.650     6.643     6.640     6.637     6.636
       14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
CV        6.654     6.654     6.653     6.653     6.653     6.653     6.653
adjCV     6.639     6.639     6.638     6.638     6.638     6.638     6.638

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X       31.74    39.44    43.72    49.90    55.77    60.46    63.26    66.58
crim    36.32    43.72    45.55    46.43    46.80    46.97    47.16    47.24
      9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X       70.01     72.74     75.52      79.0     82.42     84.62     87.29
crim    47.30     47.35     47.39      47.4     47.40     47.40     47.40
      16 comps  17 comps  18 comps  19 comps  20 comps
X        90.63     92.34     97.50     98.74    100.00
crim     47.41     47.41     47.41     47.41     47.41
```

In this case the lowest CV value (1.975) was obtained with 14 directions

- **lasso**

```r
lasso_fit <- cv.glmnet(x_boston[train_boston,], y_boston[train_boston], alpha = 1)
```

(f) Compare the effectiveness of each model on training vs. testing data. Which model is the best?

- **Multiple linear model**

```r
pred_lm <- predict(multiple_lm, Boston[test_boston,])

lm_train_error <- summary(multiple_lm)$sigma^2
lm_test_error <- mean((pred_lm - y_test)^2)


print(paste0("Train error:",round(lm_train_error,3)))
```

```
[1] "Train error:42.8"
```

```r
print(paste0("Test error:",round(lm_test_error,3)))
```

```
[1] "Test error:37.611"
```

- **Principal Components Regression**

```r
pcr_pred <- predict(pcr_fit_crime, Boston[test_boston,], ncomp = 19)

pcr_train_error <- mean(pcr_fit_crime$residuals^2)
pcr_test_error <- mean((pcr_pred - y_test)^2)
```

```r
print(paste0("Train error:",round(pcr_train_error,3)))
```

```
[1] "Train error:44.704"
```

```r
print(paste0("Test error:",round(pcr_test_error,3)))
```

```
[1] "Test error:38.48"
```

- **Partial Least Squares**

```r
pls_pred <- predict(pls_fit_crime, Boston[test_boston,], ncomp = 14)

pls_train_error <- mean(pls_fit_crime$residuals^2)
pls_test_error <- mean((pls_pred - y_test)^2)


print(paste0("Train error:",round(pls_train_error,3)))
```

```
[1] "Train error:41.317"
```

```r
print(paste0("Test error:",round(pls_test_error,3)))
```

```
[1] "Test error:37.584"
```

- **lasso**

```r
best_lambda <- lasso_fit$lambda.min
lasso_pred_train <- predict (lasso_fit , s = best_lambda , newx = x_boston[train_boston,])
lasso_pred_test <- predict (lasso_fit , s = best_lambda , newx = x_boston[test_boston,])

lasso_train_error <- mean((lasso_pred_train - y_train)^2)
lasso_test_error <- mean((lasso_pred_test - y_test)^2)


print(paste0("Train error:",round(lasso_train_error,3)))
```

```
[1] "Train error:40.677"
```

```r
print(paste0("Test error:",round(lasso_test_error,3)))
```

```
[1] "Test error:37.374"
```

Which model is the best? In this case, the best model with the lowest MSE is the lasso model (MSE = 37.374)

(g) Refit the principal components regression model and the lasso model to the entire dataset. Comment on the differences between the two methods. (Hint: also pay attention to highly correlated variables that you found in part (c).)

- **Principal Components Regression full dataset**

```r
pcr_fit_full_crime <- pcr(crim ~ .,
                 scale = TRUE,
                 validation = "CV",
                 data = Boston )

summary(pcr_fit_full_crime)
```

```
Data:   X dimension: 506 20
```

```
    Y dimension: 506 1
Fit method: svdpc
Number of components considered: 20

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV            8.61    7.148    7.144    6.808    6.727    6.724    6.719
adjCV         8.61    7.146    7.142    6.801    6.724    6.721    6.716
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       6.711    6.691    6.664     6.674     6.701     6.695     6.694
adjCV    6.708    6.686    6.661     6.669     6.695     6.689     6.687
       14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
CV        6.688     6.619     6.621     6.629     6.574     6.566     6.521
adjCV     6.674     6.609     6.611     6.619     6.563     6.555     6.511

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X       31.96    42.15    49.53    56.18    61.90    67.43    72.79    77.66
crim    31.36    31.49    38.18    39.63    39.74    39.83    39.95    40.37
      9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X       82.19     86.38     89.81     92.54     94.42     95.67     96.88
crim    40.85     41.12     41.31     41.49     41.64     42.52     43.74
      16 comps  17 comps  18 comps  19 comps  20 comps
X        97.88     98.79     99.43     99.82    100.00
crim     43.76     43.94     45.12     45.26     46.04
```

- **lasso full dataset**

```r
lasso_fit_full <- cv.glmnet(x_boston, y_boston, alpha = 1)
best_lambda_full <- lasso_fit_full$lambda.min
lasso_pred_full <- predict(lasso_fit_full , s = best_lambda_full,
                           newx = x_boston)
```

```r
# MSE comparison Lasso vrs PCR

# MSE PCR
(pcr_full_data_error <- mean(pcr_fit_full_crime$residuals^2))
```

```
[1] 43.664
```

```r
# MSE Lasso
(lasso_full_data_error <- mean((lasso_pred_full - y_boston)^2))
```

```
[1] 39.98714
```

###Comment on the differences between the two methods. (Hint: also pay attention to highly correlated variables that you found in part (c)

PCR is an approach that can be useful when the number of variables is a lot greater than the number of observations (p » n) and when the variables are highly correlated between each other. With this approach it is possible to reduce the number of variables (dimension reduction) to a few M components that are independent from each other that can expalin the variability in the dataset.

The main difference between the PCR and the LASSO is the latter can perform variable selection which can facilitate model interpretation, while PCR only performs dimention reduction.

(h) Refit the partial least squares model to the entire dataset, and compare with the principal components regression model.

```r
pls_fit_full_crime <- plsr(crim ~ .,
                     scale = TRUE,
                     validation = "CV",
                     data = Boston)
summary(pls_fit_crime)
```

```
Data:   X dimension: 404 20
    Y dimension: 404 1
Fit method: kernelpls
Number of components considered: 20

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           8.805    7.095    6.752    6.759    6.721    6.703    6.680
adjCV        8.805    7.091    6.745    6.742    6.705    6.686    6.665
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       6.656    6.667    6.665     6.658     6.655     6.652     6.651
adjCV    6.642    6.651    6.650     6.643     6.640     6.637     6.636
       14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
CV        6.654     6.654     6.653     6.653     6.653     6.653     6.653
adjCV     6.639     6.639     6.638     6.638     6.638     6.638     6.638

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X       31.74    39.44    43.72    49.90    55.77    60.46    63.26    66.58
crim    36.32    43.72    45.55    46.43    46.80    46.97    47.16    47.24
      9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X       70.01     72.74     75.52      79.0     82.42     84.62     87.29
crim    47.30     47.35     47.39      47.4     47.40     47.40     47.40
      16 comps  17 comps  18 comps  19 comps  20 comps
X        90.63     92.34     97.50     98.74    100.00
crim     47.41     47.41     47.41     47.41     47.41
```

```r
# MSE comparison PLS vrs PCR

# MSE PCR
(pcr_full_data_error <- mean(pcr_fit_full_crime$residuals^2))
```

```
[1] 43.664
```

```r
# MSE Lasso
(pls_full_data_error <- mean(pls_fit_full_crime$residuals^2))
```

```
[1] 40.52833
```

When compared, PLS have lower MSE than PCR