# Problem Set 3: OLS Regression & Resampling

Stat 154, Spring 2018

*Due date: Fri Mar-02 (before midnight)*

## 1) Hat Matrix (30 pts)

Let $\mathbf{X}$ be an $n \times p$ matrix of full (column) rank, and let $\mathbf{H} = \mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}$.

a. Show that $\mathbf{H}$ is symmetric. (10 pts)
b. Show that $\mathbf{H}$ is idempotent. (10 pts)
c. Consider the matrix $\mathbf{Q} = \mathbf{I} - \mathbf{H}$. Show that $\mathbf{Q}$ is symmetric and idempotent. (10 pts)

## 2) Eigenstructure of Idempotent Matrices (10 pts)

Show that the eigenvalues of idempotent matrices are always either zeros or ones.

## 3) Auxiliary Functions (10 pts)

a. Create a function `R2()` that takes an object of class `"lm"` and returns the coefficient of determination $R^2$. You are NOT allowed to use `summary()`.

Test your function `R2()` with the following code. You may want to compare your result with the one provided by `summary(lm_obj)$r.squared`.

```r
# lm object
lm_obj <- lm(mpg ~ disp + hp + wt, data = mtcars)

# coeff of determination
R2(lm_obj)

# compare to
summary(lm_obj)$r.squared
```

b. Create a function `AR2()` that takes an object of class `"lm"` and returns the adjusted coefficient of determination $R^2_{adj}$. Use your function `R2()` inside the body of `AR2()`. You are NOT allowed to use `summary()`.

Test your function `AR2()` with the following code. You may want to compare your result with the one provided by `summary(lm_obj)$adj.r.squared`

```
# lm object
lm_obj <- lm(mpg ~ disp + hp + wt, data = mtcars)

# coeff of determination
AR2(lm_obj)

# compare to
summary(lm_obj)$adj.r.squared
```

c. Create a function `MSE()` that takes an object of class `"lm"` and returns the mean squared error (i.e. training MSE), basically: MSE $= (1/n) \sum (y_i - \hat{y}_i)^2$. You are NOT allowed to use `summary()`.

Test your function `MSE()` with the following code:

```
# lm object
lm_obj <- lm(mpg ~ disp + hp + wt, data = mtcars)

# mean squared error
MSE(lm_obj)
```

## 4) Bike Sharing: Fitting Models (20 pts)

In this problem you will work with the *Bike Sharing Data Set* (courtesy of Hadi Fanaee) that contains the hourly and daily count of rental bikes between years 2011 and 2012 in Capital bikeshare system with the corresponding weather and seasonal information. Visit the following website for more information:

https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

All the data files are in a zip file. This file contains a `Readme.txt` file, and two CSV data files: `day.csv` and `hour.csv`. One option to download the zip file, and then extract its contents, is using the R functions `download.file()` and `unzip()`—of course, you can use other approaches to get the data.

```
# =========================================================
# do NOT include the code in this chunk in your Rmd file!!!
# =========================================================
# assembling url
uci <- 'https://archive.ics.uci.edu/ml/machine-learning-databases/'
zip <- '00275/Bike-Sharing-Dataset.zip'
url <- paste0(uci, zip)

# download zip file, and unzip its contents in working directory
```

```r
download.file(url, 'Bike-Sharing-Dataset.zip')
unzip('Bike-Sharing-Dataset.zip')
```

### Motivation

OLS Regression analysis can be used to answer the following question:

> *What is the predicted number of riders for a Sunday that is expected to be sunny and cool (around 75 F degrees)?*

We may try to predict ridership, given the weather conditions, day of the week, time of the year and so on. This type of prediction could actually be useful for bike-sharing services that may need to prepare for days with high demand.

Using the daily data set (i.e. file `day.csv`), the idea is to fit various models of different complexity and examine some of their outputs to assess their predictive performance.

### 4.1) Data Processing

- Import the file `day.csv`
- Subset those observations of `yr == 0` (i.e. year 2011).
- Create a new binary variable `clearday` by selecting `weathersit == 1`. In other words, values with `weathersit == 1` should be assigned to one, while the rest should be assigned to zero.
- *Note*: temperature, `temp`, is scaled to have values in a range [0, 1]

$$\frac{\text{Celsius temperature} - \text{minimum}}{\text{maximum} - \text{minimum}}$$

where the min and max were -8 and 39, respectively.

### 4.2) Fitting models (10 pts)

Use `lm()` to fit the following regression models to predict bike ridership (i.e. `registered`) using a handful of predictors. As you will notice, the complexity of the models will increase from one regression to the next one.

**Model 1**

$$\texttt{registered} = \beta_0 + \beta_1 \texttt{temp} + \varepsilon$$

**Model 2**

$$\texttt{registered} = \beta_0 + \beta_1\texttt{temp} + \beta_2\texttt{temp}^2 + \varepsilon$$

**Model 3**

$$\texttt{registered} = \beta_0 + \beta_1\texttt{temp} + \beta_2\texttt{temp}^2 + \beta_3\texttt{workingday} + \varepsilon$$

**Model 4**

$$\texttt{registered} = \beta_0 + \beta_1\texttt{temp} + \beta_2\texttt{temp}^2 + \beta_3\texttt{workingday} + \beta_4\texttt{clearday} + \varepsilon$$

**Model 5**

$$\texttt{registered} = \beta_0 + \beta_1\texttt{temp} + \beta_2\texttt{temp}^2 + \beta_3\texttt{workingday} + \beta_4\texttt{clearday} + \beta_5(\texttt{temp} \times \texttt{workingday}) + \varepsilon$$

### 4.3) Basic Models Comparison (10 pts)

Use your functions `MSE()`, `R2()`, and `AR2()` to create a table (e.g. data frame) `model_quality` for these "quality" measures (MSE, $R^2$ and $R^2_{adj}$), of all the five fitted models. Use the rows for the models, and the columns for the quality measures.

Also, plot the (training) MSEs against the number of regressor terms in each model. We will treat the number of regressor terms as the "complexity" of the models.

- Which model has the smallest in-sample MSE?
- Do you observe any trend in the graph?
- You don't need to include an answer for these questions, just answer them mentally.

## 5) Bike Sharing: Hold-Out method (20 pts)

As we mentioned in class, the *training MSEs* are not an appropriate way to assess the predictive power of the models, since the training set, which is used for calibrating models, is also used for model testing.

The training (or in-sample) MSEs tend to have an optimistic bias towards complicated models. There are in-sample metrics that take model complexity into account, such as Akaike's information criterion (AIC) and Bayesian Information Criterion (BIC), but these are not applicable when an explicit likelihood is not present in the model.

The most straightforward approach is to split the dataset into two parts: one for training and one for testing. This approach is commonly known as the *holdout method*. A common split is 80-20: use 80% of the data to train the model and 20% to test the model.

- Select 20% of your dataset as holdout. You should use simple random sampling. Before generating the sample, specify a random seed with `set.seed()`—for reproducibility purposes.
- For each of the regression models in *Problem 4*, train on the remaining 80% of the data, predict the holdout data, and compute the **test MSE**. Identify which model gives the lowest holdout test MSE.

## 6) Bike Sharing: Cross-validation (20 pts)

*Cross-validation* is an alternative to the holdout method. A useful package for such a task is `"caret"`. To generate folds, we can use `createFolds()`.

```r
library(caret)
# list with 5-folds
folds <- createFolds(mtcars$mpg, k = 5)
folds
```

```
## $Fold1
## [1]  1  5  9 19 20 22 31
##
## $Fold2
## [1] 11 14 17 18 21 29 32
##
## $Fold3
## [1]  2  3  7  8 13 23
##
## $Fold4
## [1]  6 16 25 27 30
##
## $Fold5
## [1]  4 10 12 15 24 26 28
```

By default, ten folds are generated. Note that `folds` contains a list of vectors of indices. Since randomness is involved, you might get a different list. This is why you should also specify a random seed with `set.seed()` so you can replicate your analyses.

### 6.1) 10-Folds and test MSEs (10 pts)

Use `createFolds()` to create a list `folds` to do a 10-fold cross validation to estimate the prediction error for bike ridership. Specifically,

- For each fold,
  - For each regression model,
    - Train the model based on all observations except the ones in the fold.

    * Predict the observation in the fold.
    * Compute the test MSE of the predictions.

You should end up having a $5 \times 10$ matrix with rows corresponding to the models and columns corresponding to the folds.

### 6.2) CV-MSE (10 pts)

The CV-MSE is defined as:

$$\text{MSE}_{CV} = \frac{1}{\text{number of folds}} \sum_{\text{fold}} \text{MSE}_{\text{fold}}$$

which is simply the average MSE over the folds.

1. Calculate the CV-MSE for each model.
2. Plot the CV-MSEs against the order of the regression models.
3. Which model gives the lowest CV-MSEs? Is it reasonable? Why or why not?

## 7) Bike Sharing: Bootstrap (20 pts)

*Bootstrap* is another popular approach for model assessment. The idea is to iterate the following procedure many times: first, sample with replacement from the data (this serves as the training set); second, train the model on the sampled data; third, test the model on the data that is not in the sample and compute the performance metric, typically the MSE for regression problems. Finally, compute the average MSE over all the iterations, and this is referred as the *bootstrap MSE*.

Do the following tasks with 400 bootstrap samples.

1. Plot the bootstrap MSEs against each regression model.
2. Which model gives the lowest bootstrap MSEs? Is it reasonable? Why or why not?
3. For each model, compute the SD of the 400 MSEs. Plot the SD against the model complexity. What do you notice?
4. For each model, make a histogram of the 400 MSEs. What do you notice?