# Homework 1

*STAT 430, Spring 2017*

*Due: Friday, February 3 by 11:59 PM*

## Exercise 1

```
library(tibble)
library(readr)

make_hw01_data = function(n_obs = 1000) {

  a = runif(n = n_obs, min = 0, max = 3)
  b = runif(n = n_obs, min = 0, max = 5)
  c = runif(n = n_obs, min = 0, max = 1)
  d = rbinom(n = n_obs, size = 1, p = 0.5)

  eps = rnorm(n = n_obs, mean = 0 , sd = 1)

  y = -5 + 3 * a ^ 2 + 3 * b + 2.5 * b * d + eps

  tibble(y, a, b, c, d)

}

set.seed(42)
hw01_data = make_hw01_data()
write_csv(hw01_data, "hw01-data.csv")
```

[**12 points**] This question will use data in a file called `hw01-data.csv`. The data contains four predictors `a`, `b`, `c`, `d`, and a response `y`.

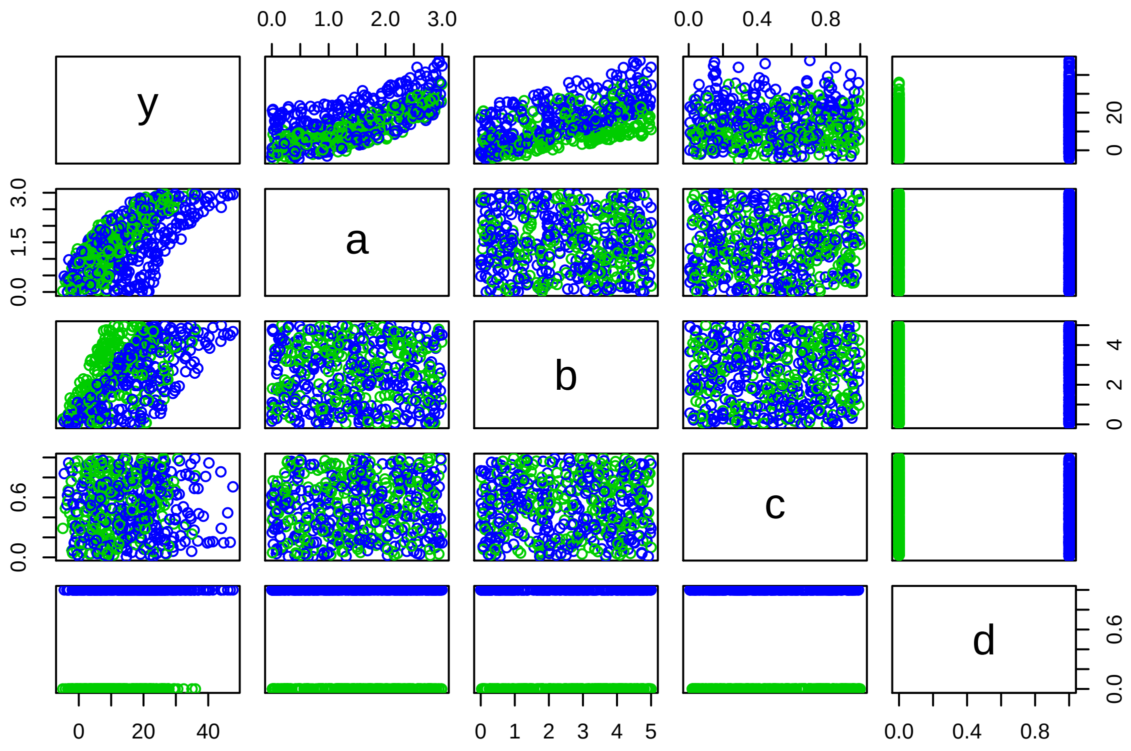
Use `set.seed(42)` to control randomization, then randomly split the data into train and test sets using half of the data for each. Next, fit four models using the training data:

- Model 1: `y ~ .`
- Model 2: `y ~ . + I(a ^ 2) + I(b ^ 2) + I(c ^ 2)`
- Model 3: `y ~ . ^ 2 + I(a ^ 2) + I(b ^ 2) + I(c ^ 2)`
- Model 4: `y ~ a * b * c * d * I(a ^ 2) * I(b ^ 2) * I(c ^ 2)`

```
hw01_data = read_csv("hw01-data.csv")
```

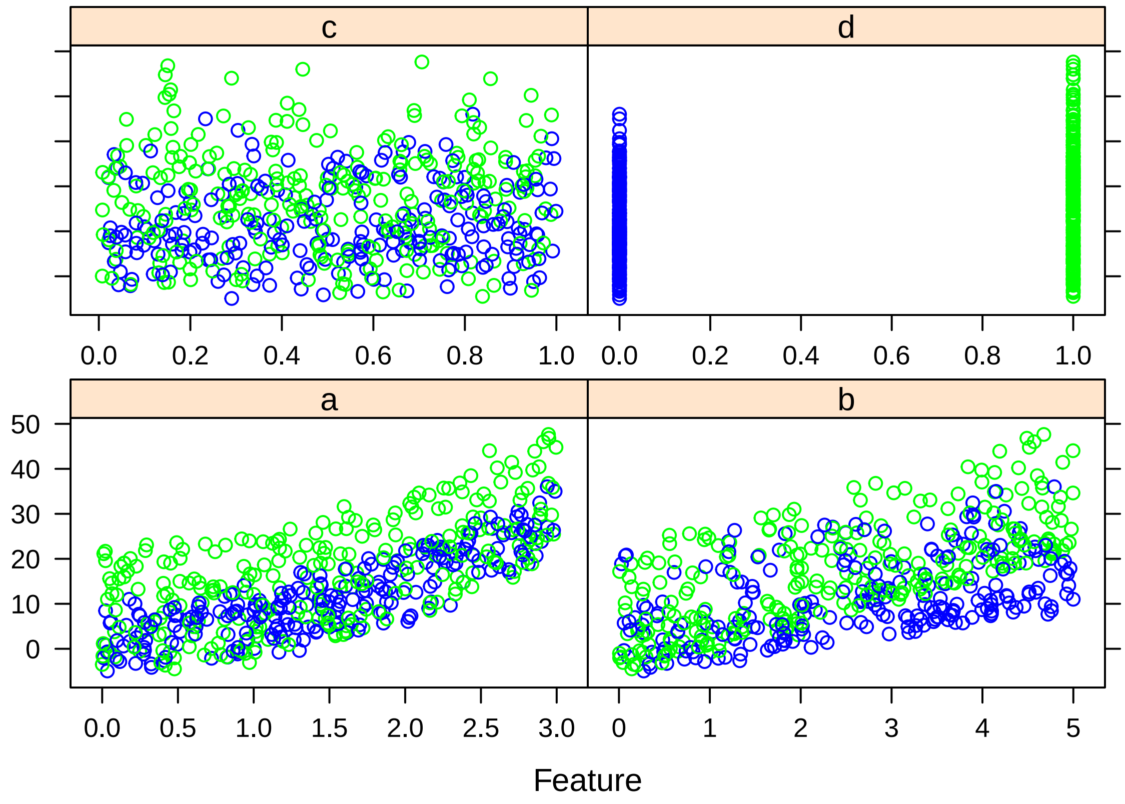
```
set.seed(42)
train_index = sample(1:nrow(hw01_data), size = round(0.5 * nrow(hw01_data)))
train_data = hw01_data[train_index, ]
test_data = hw01_data[-train_index, ]
```

```
pairs(train_data, col = train_data$d + 3)
```



Instead of `pairs()` which also shows the relationships among the predictors, we could instead use the `featurePlot()` function from the `caret` package.

```r
library(caret)
featurePlot(x = train_data[, c("a", "b", "c", "d")], y = train_data$y,
            col = ifelse(train_data$d, "Green", "Blue"))
```



```r
fit1 = lm(y ~ ., data = train_data)
fit2 = lm(y ~ a + b + c + d + I(a ^ 2) + I(b ^ 2) + I(c ^ 2), data = train_data)
fit3 = lm(y ~ . ^ 2 + I(a ^ 2) + I(b ^ 2) + I(c ^ 2), data = train_data)
fit4 = lm(y ~ a * b * c * d * I(a ^ 2) * I(b ^ 2) * I(c ^ 2), data = train_data)
fit5 = lm(y ~ a + b * d + I(a ^ 2), data = train_data)
```

**(a)** For each of the models above, report:

- Train RMSE
- Test RMSE
- Number of Parameters, Excluding the Variance

To receive full marks, arrange this information in a well formatted table.

**Solution:**

```r
model_list = list(fit1, fit2, fit3, fit4, fit5)

train_rmse = sapply(model_list, get_rmse, data = train_data, response = "y")
test_rmse = sapply(model_list, get_rmse, data = test_data, response = "y")
num_params = sapply(model_list, get_num_params)
```

**Comments**: The results can be seen in the table below. Note that there is also a `fit5` which is used later. Be aware that the code to create the table below can be found in the accompanying `.Rmd` file, which also includes some helper functions written to aide in creating the numerical results.

| Model | Train RMSE | Test RMSE | Parameters |
|---|---|---|---|
| `fit1` | 2.9954566 | 2.9487898 | **5** |

| Model | Train RMSE | Test RMSE | Parameters |
|---|---|---|---|
| `fit2` | 2.0837849 | 2.192281 | 8 |
| `fit3` | 1.0211239 | *1.0413432* | 14 |
| `fit4` | **0.8979202** | 1.2718111 | 128 |
| `fit5` | 1.0277506 | **1.0345291** | 6 |

**(b)** Based on these results do you have evidence that any of these models are overfitting or underfitting? If so, which models?

**Solution:**

**Comments**: We would believe the `fit1` is underfitting since it does not have the lowest test RMSE and has the least parameters, all while having the highest train RMSE. That is, it is an inflexible model that does not fit well. `fit2` also fits this description, only with a slightly better train RMSE. `fit4` is clearly overfitting as it has the lowest train RMSE, but not the lowest test RMSE, while having the most parameters.

**(c)** Find a model that outperforms each of the models above. Report this model's train RMSE, test RMSE, and number of parameters used. **Hint:** If you haven't already, consider some exploratory data analysis. **Hint:** Your instructor's solution uses a model with only six parameters. Yours may have more.

**Solution:**

**Comments**: See `fit5` on the table above. It is the model `y ~ a + b * d + I(a ^ 2)` which is rather small compared to some of the others.

Some justification for this model can be seen above in the `pairs()` plot. First, we see that there is a curved relationship between `y` and `a`. Also notice that `d` is essentially a dummy variable, and to look for interactions, we have colored all points according to this variable. We see a rather obvious interaction in the relationship between `y` and `b`. The slope is noticeably different for different values of `d`. There seems to be no relationship between `y` and `c`.

Also note, you can find the code that generated this data in the `.Rmd`, which shows that this is actually the **best** possible model.

## Exercise 2

[**8 points**] For this question we will use the `Boston` data from the `MASS` package. Use `?Boston` to learn more about the data.

```r
library(MASS)
data(Boston)
Boston = as_tibble(Boston)
```

Use `set.seed(314)` to control randomization, then randomly split the data into train and test sets using 456 observations for the training data and the remainder for the testing data. (Roughly 10 percent of the data for the test set.)

Fit a (potentially large) number of **nested** linear models with `medv` as the response. Use train and test RMSE to determine: two models that are probably underfitting, two models that are probably overfitting, and one model between the under and overfitting models. Report (only) these five models as well as their train RMSE, test RMSE, and number of parameters. Note: you may report the models used using their `R` syntax. To receive full marks, arrange this information in a well formatted table.

**Solution:**

```r
set.seed(314)
train_index = sample(1:nrow(Boston), size = 456)
train_data = Boston[train_index, ]
test_data = Boston[-train_index, ]
```

```r
fit1 = lm(medv ~ crim, data = train_data)
fit2 = lm(medv ~ ., data = train_data)
fit3 = lm(medv ~ . ^ 2, data = train_data)
fit4 = lm(medv ~ . ^ 2 + I(crim ^ 2), data = train_data)
fit5 = lm(medv ~ . ^ 2 + I(crim ^ 2) + I(lstat ^ 2) + I(rm ^ 2), data = train_data)
```

```r
model_list = list(fit1, fit2, fit3, fit4, fit5)

train_rmse = sapply(model_list, get_rmse, data = train_data, response = "medv")
test_rmse = sapply(model_list, get_rmse, data = test_data, response = "medv")
num_params = sapply(model_list, get_num_params)
```

| Model | Train RMSE | Test RMSE | Parameters |
|---|---|---|---|
| `fit1` | 8.2078215 | 10.6461725 | **2** |
| `fit2` | 4.6209812 | 5.2700244 | 14 |
| `fit3` | 2.5308682 | **3.6871058** | 92 |
| `fit4` | 2.5102649 | 4.7790324 | 93 |
| `fit5` | **2.4896438** | 4.6394577 | 95 |

**Comments**: By looking at test RMSE, we see that `fit3` is the best model. Based on train RMSE, test RMSE, and model size relative to the model, `fit1` and `fit2` are underfitting, while `fit4` and `fit5` are overfitting.

Note, this result is **highly** dependent on the seed used. We will see how to overcome this obstacle later using resampling techniques.

## Exercise 3

[**10 points**] How do outliers and influential points affect prediction? Usually when fitting regression models for explanation, dealing with outliers is a complicated issue. When considering prediction, we can empirically determine what to do.

Continue using the Boston data, training split, and models from Exercise 2. Consider your best model from Exercise 2. Refit this model with each of the following modifications:

- Removing observations from the training data with (absolute value of) studentized residuals greater than 2.
- Removing observations from the training data with (absolute value of) studentized residuals greater than 3.
- Removing observations from the training data considered influential. That is, with a Cook's distance greater than $\frac{4}{n}$. Here, $n$ is the number of observations used to train the model.

**(a)** Use these four models, including the original model fit to unmodified data, to obtain test RMSE. Summarize these results in a table. Include the number of observations removed for each. Which performs the best? Were you justified modifying the training data?

**Solution:**

```r
train_outliers_2 = subset(train_data, abs(studres(fit3)) < 2)
train_outliers_3 = subset(train_data, abs(studres(fit3)) < 3)
train_influential = subset(train_data, cooks.distance(fit3) < 4 / nrow(train_data))
```

```r
fit3o = lm(medv ~ . ^ 2, data = train_data)
fit3a = lm(medv ~ . ^ 2, data = train_outliers_2)
fit3b = lm(medv ~ . ^ 2, data = train_outliers_3)
fit3c = lm(medv ~ . ^ 2, data = train_influential)
```

```r
model_list = list(fit3o, fit3a, fit3b, fit3c)
test_rmse = sapply(model_list, get_rmse, data = test_data, response = "medv")
```

| Dataset | Fitted Model | Test RMSE | Observations Removed |
|---|---|---|---|
| Full Training | `fit3o` | 3.6871058 | 0 |
| Studentized > 2 Removed | `fit3a` | 3.9889266 | 23 |
| Studentized > 3 Removed | `fit3b` | 3.9477735 | 8 |
| Influential Removed | `fit3c` | 4.6354554 | 28 |

**Comments**: Here we see that the original model still performs best, so we should not remove any data.

**(b)** Using the best of these fitted models, create a 99% **prediction interval** for an observation with the following values:

| crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | black | lstat |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.02763 | 75.0 | 2.95 | 0 | 0.4280 | 6.595 | 21.8 | 5.4011 | 3 | 252 | 18.3 | 395.63 | 4.32 |

**Solution:**

```r
new_data = data.frame(
  crim = 0.02763,
  zn = 75.0,
  indus = 2.95,
  chas = 0,
  nox = 0.4280,
  rm = 6.595,
  age = 21.8,
  dis = 5.4011,
  rad = 3,
  tax = 252,
  ptratio = 18.3,
  black = 395.63,
  lstat = 4.32
)
```

```r
predict(fit3o, new_data, interval = "prediction", level = 0.99)
```

```
##       fit     lwr      upr
## 1 28.8457 20.8359 36.85551
```