Homework 01

STAT 430, Fall 2017

Due: Friday, September 15, 11:59 PM

Exercise 1

```
library(tibble)
library(readr)

make_hw01_data = function(n_obs = 1000) {

  a = runif(n = n_obs, min = 0, max = 2)
  b = runif(n = n_obs, min = 0, max = 2)
  c = runif(n = n_obs, min = 0, max = 2)
  d = rbinom(n = n_obs, size = 1, p = 0.5)
  eps = rnorm(n = n_obs, mean = 0 , sd = 0.5)
  y = -5 + 3 * a ^ 2 + 4 * c + 3.5 * c * d + eps
  tibble(y, a, b, c, d)

}

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

[**10 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`.

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

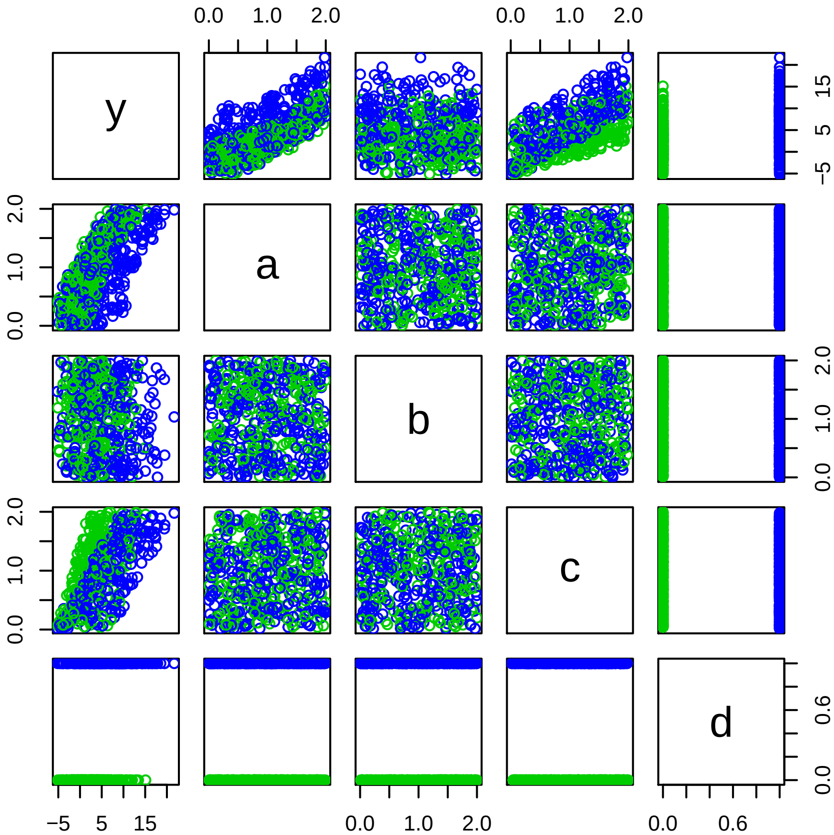
After reading in the data as `hw01_data`, use the following code to test-train split the data.

```
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, ]
```

Next, fit four linear models using the training data:

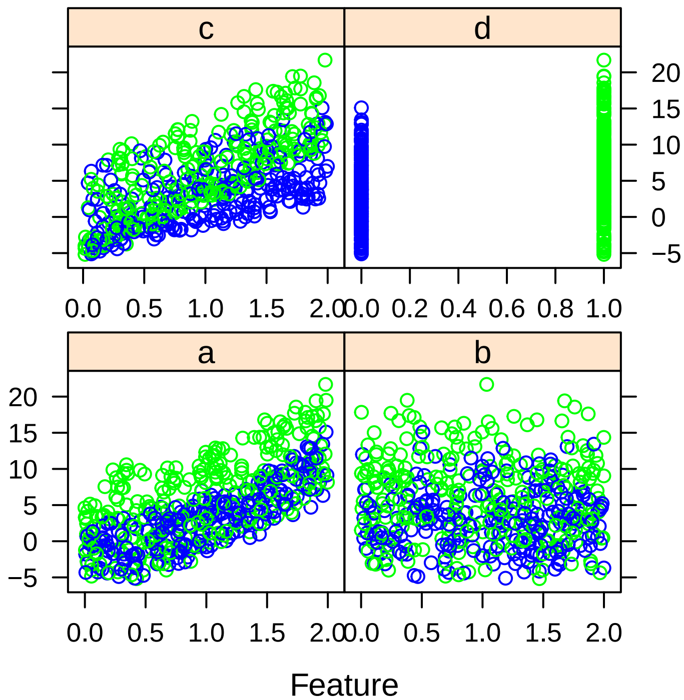
- Model 1: `y ~ .`
- Model 2: `y ~ . + I(a ^ 2) + I(b ^ 2) + I(c ^ 2)`
- Model 3: `y ~ . ^ 2 + I(a ^ 2) + I(b ^ 2) + I(c ^ 2)`
- Model 4: `y ~ a * b * c * d + I(a ^ 2) + I(b ^ 2) + I(c ^ 2)`

```
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.

```
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 + c * d + I(a ^ 2), data = train_data)
```

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. Also note which model is best for making predictions.

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`	1.4381782	1.4286911	**5**
`fit2`	1.1242482	1.1526319	8
`fit3`	0.5105619	*0.5206716*	14
`fit4`	**0.5082713**	0.5211251	19
`fit5`	0.5138062	**0.5164304**	7

Based on these results, **Model 3** is the best model for prediction. (`fit5` was not required.)

[**Not Graded**] For fun, find a model that outperforms each of the models above. *Hint:* Consider some exploratory data analysis. *Hint:* Your instructor's solution uses a model with only seven parameters. Yours may have more.

Solution:

Comments: See `fit5` on the table above. It is the model `y ~ a + c * 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 `c` and `d`. The slope is noticeably different for different values of `d`. You might think the same applies to `a` and `d`, however that is only a shift, not a change in shape (slope or curve). The shift is taken care of by including `d` in the model. There seems to be no relationship between `y` and `b`.

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

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

```
library(readr)
library(tibble)
library(MASS)
data(Boston)
Boston = as_tibble(Boston)
```

Use the following code to test-train split the data.

```
set.seed(42)
boston_index = sample(1:nrow(Boston), size = 400)
train_boston = Boston[boston_index, ]
test_boston  = Boston[-boston_index, ]
```

Fit the following linear model that uses `medv` as the response.

```
fit = lm(medv ~ . ^ 2, data = train_boston)
```

Fit two additional models, both that perform worse than `fit`, with respect to prediction. One should be a smaller model, relative to `fit`. The other should be a larger model, relateive to `fit`. Call them `fit_smaller` and `fit_larger` respectively. Any "smaller" model should be nested in any "larger" model.

Report these three 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:

```
fit_smaller = lm(medv ~ crim, data = train_boston)
fit = lm(medv ~ . ^ 2, data = train_boston)
fit_larger  = lm(medv ~ . ^ 2 + I(crim ^ 2) + I(lstat ^ 2) + I(rm ^ 2), data = train_boston)
```

```r
model_list = list(fit_smaller, fit, fit_larger)

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

- `fit_smaller`: `medv ~ crim`
- `fit`: `medv ~ . ^ 2`
- `fit_larger`: `medv ~ . ^ 2 + I(crim ^ 2) + I(lstat ^ 2) + I(rm ^ 2)`

Model	Train RMSE	Test RMSE	Parameters
`fit_smaller`	8.2541439	9.22791	2
`fit`	2.6239518	3.2202003	92
`fit_larger`	2.5995498	3.4295235	95

Exercise 3

[**10 points**] How do outliers 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 the model stored in `fit` from Exercise 2. Obtain the standardized residuals from this fitted model. Refit this model with each of the following modifications:

- Removing observations from the training data with absolute standardized residuals greater than 2.
- Removing observations from the training data with absolute standardized residuals greater than 3.

(a) Use these three fitted 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_boston, abs(rstandard(fit)) < 2)
train_outliers_3 = subset(train_boston, abs(rstandard(fit)) < 3)
```

```r
fit   = lm(medv ~ . ^ 2, data = train_boston)
fit_2 = lm(medv ~ . ^ 2, data = train_outliers_2)
fit_3 = lm(medv ~ . ^ 2, data = train_outliers_3)
```

```r
model_list = list(fit, fit_2, fit_3)
test_rmse = sapply(model_list, get_rmse, data = test_boston, response = "medv")
```

Dataset	Fitted Model	Test RMSE	Observations Removed
Full Training	`fit`	3.2202003	0
Std. Resid $>$ 2 Removed	`fit_2`	3.0088003	19
Std. Resid $>$ 3 Removed	`fit_3`	3.0560078	5

Comments: Based on these results, we do see justification for removing outliers. Both strategies see improvement, but more improvement with more (smaller) outliers excluded when fitting.

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

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
0.02763	75.0	3.95	0	0.4280	6.595	22.8	5.4011	3	252	19.3	395.63	4.32

Solution:

```
new_data = data.frame(
  crim = 0.02763,
  zn = 75.0,
  indus = 3.95,
  chas = 0,
  nox = 0.4280,
  rm = 6.595,
  age = 22.8,
  dis = 5.4011,
  rad = 3,
  tax = 252,
  ptratio = 19.3,
  black = 395.63,
  lstat = 4.32
)
```

```
predict(fit_2, new_data, interval = "prediction", level = 0.99)
```

```
##        fit      lwr      upr
## 1 27.52639 21.03786 34.01491
```