# Homework3_Zixin Ouyang

*Zixin Ouyang*

*9/29/2017*

**Exercise 1**

```
library(FNN)
```

```
hw03_train<-read.csv('hw03-train-data.csv')
hw03_test<-read.csv('hw03-test-data.csv')
```

```
rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
```

```
make_knn_pred = function(k = 1, training, predicting) {
  pred = FNN::knn.reg(train = training[c('x1','x2','x3','x4')],
                      test = predicting[c('x1','x2','x3','x4')],
                      y = training$y, k = k)$pred
  act  = predicting$y
  rmse(predicted = pred, actual = act)
}
```

```
k = c(1,5,25)
```

```
knn_tst_rmse = sapply(k, make_knn_pred,
                      training = hw03_train,
                      predicting = hw03_test)
```

```
scaled_train=data.frame(hw03_train['y'],sapply(hw03_train[c('x1','x2','x3','x4')],scale))
scaled_test=data.frame(hw03_test['y'],sapply(hw03_test[c('x1','x2','x3','x4')],scale))
```

```
scaled_tst_rmse = sapply(k, make_knn_pred,
                      training = scaled_train,
                      predicting = scaled_test)
```

| K | Test RMSE | Scaling Used |
|---|---|---|
| 1 | 0.6839884 | No |
| 1 | 0.7214409 | Yes |
| 5 | 0.5369142 | No |
| 5 | 0.5467248 | Yes |
| 25 | 0.5157672 | No |
| 25 | 0.5081259 | Yes |

**Exercise 2**

```
library(ISLR)
auto = Auto[, !names(Auto) %in% c("name")]
```

```r
set.seed(42)
auto_idx = sample(1:nrow(auto), size = round(0.5 * nrow(auto)))
auto_trn = auto[auto_idx, ]
auto_tst = auto[-auto_idx, ]
```

```r
mod_1=lm(mpg ~ ., data = auto_trn)
lm_rmse=rmse(auto_tst$mpg, predict(mod_1, newdata=auto_tst))
```

```r
mod2_pred=FNN::knn.reg(train =sapply(auto_trn[,c(2,3,4,5,6,7,8)],scale),
                       test = sapply(auto_tst[,c(2,3,4,5,6,7,8)],scale),
                       y = auto_trn$mpg, k = 10)
knn_rmse=rmse(auto_tst$mpg, mod2_pred$pred)
```

The test RMSE for the additive linear model is 3.0684888, while the test RMSE for the k-nearest neighbors model is 2.9932617. The vaue of k used is 10, and I have scaled the X data.

### Exercise 3

```r
f = function(x) {
  x ^ 2
}
```

```r
get_sim_data = function(f, sample_size = 100) {
  x = runif(n = sample_size, min = 0, max = 1)
  y = rnorm(n = sample_size, mean = f(x), sd = 0.3)
  data.frame(x, y)
}
```

```r
set.seed(659017838)
n_sims = 500
n_models = 3
x = data.frame(x = 0.90)
predictions = matrix(0, nrow = n_sims, ncol = n_models)
```

```r
for(sim in 1:n_sims) {
  sim_data = get_sim_data(f)
  fit_1 = FNN::knn.reg(train = sim_data['x'], test = x, y = sim_data$y, k = 1)
  fit_10 = FNN::knn.reg(train = sim_data['x'], test = x, y = sim_data$y, k = 10)
  fit_100 = FNN::knn.reg(train = sim_data['x'], test = x, y = sim_data$y, k = 100)

  predictions[sim, 1] = fit_1$pred
  predictions[sim, 2] = fit_10$pred
  predictions[sim, 3] = fit_100$pred

}
```

```r
get_mse = function(truth, estimate) {
  mean((estimate - truth) ^ 2)
}
```

```r
get_bias = function(estimate, truth) {
  mean(estimate) - truth
}
```

```r
get_var = function(estimate) {
  mean((estimate - mean(estimate)) ^ 2)
}
```

```r
bias = apply(predictions, 2, get_bias, truth = f(x = 0.90))
variance = apply(predictions, 2, get_var)
mse = apply(predictions, 2, get_mse, truth = f(x = 0.90))
```

| K | Mean Squared Error | Bias Squared | Variance |
|---|---|---|---|
| 1 | 0.09669 | 0.00004 | 0.09665 |
| 10 | 0.00894 | 0.00005 | 0.00890 |
| 100 | 0.23040 | 0.22882 | 0.00158 |

**Exercise 4**

(a) k=25 performs best, because it has the smallest test RMSE.
(b) When k increases and test RMSE decreases, scaling data could make test RMSE smaller, so it is appropriate.
(c) Based on the exploratory data analysis, we see some non-linear relationships that an additive model would not detect, but a non-parametric method like KNN would automatically approximate.
(d) The k-nearest neighbors with k=1 and the one with k=10 are providing unbiased predictions,since their squared bias are much smaller than the model with k=100.
(e) The k-nearest neighbors with k=10 is predicting best at x=0.90, because it has the smallest RMSE and its squared bias and variance are very close to the smallest values .