Problem 3

According to my simulation, it seems that dimensionality benefits knn method if we compare the best fit using knn(from the perspective of EPE) and linear fit for the data. Since the model is very specific, it is hard to draw a general conclusion on the behaviour of knn prediction under high dimensional case. However, I feel knn can be more accurate than linear model in prediction as we increase dimension.

```
library(FNN)

## Warning:  package 'FNN' was built under R version 3.0.2

set.seed(14250)
X <- matrix(runif(250, min = 0, max = 2 * pi), nrow = 50)
f0_x <- function(x) {
    sin <- sin(x[1]) + sin(sqrt(2) * x[2]) + sin(sqrt(3) * x[3]) +
        sin(sqrt(4) * x[4]) + sin(sqrt(5) * x[5])
    cos <- sum(cos(x[1:4] * x[2:5]))
    y <- sin + cos
    return(y)
}
Y <- apply(X, 1, f0_x) + rnorm(50)  # I suppose this is the case although the homework sheet is differen
# Suppose I use one point Xo
Xo <- runif(5, min = 0, max = 2 * pi)
# Simulate one thousand time of Yo of Xo
Yo <- f0_x(Xo) * rep(1, 1000) + rnorm(1000)
# Prediction of knn using different k values.
Yo_knnfive <- knn.reg(train = X, test = Xo, y = Y, k = 5)
EPE_5 <- mean((Yo_knnfive$pred - Yo)^2)
EPE_5

## [1] 0.9692

Yo_knn10 <- knn.reg(train = X, test = Xo, y = Y, k = 10)
EPE_10 <- mean((Yo_knn10$pred - Yo)^2)
EPE_10

## [1] 1.113

Yo_knn20 <- knn.reg(train = X, test = Xo, y = Y, k = 20)
EPE_20 <- mean((Yo_knn20$pred - Yo)^2)
EPE_20

## [1] 2.041

Yo_knn30 <- knn.reg(train = X, test = Xo, y = Y, k = 30)
EPE_30 <- mean((Yo_knn30$pred - Yo)^2)
EPE_30

## [1] 2.519

# Linear Prediction
lfit <- lm(Y ~ X)
Y_hat <- Xo %*% lfit$coefficients[2:6] + lfit$coefficients[1]
EPE_linear <- mean((Y_hat - Yo)^2)
EPE_linear

## [1] 1.042
```

```r
# Now I need to get E(EPE(x)), and it is estimated by
# simulation
Expected_EPE <- function(method, X, Y, k_value = NULL) {
    X_new <- matrix(runif(10000, min = 0, max = 2 * pi), nrow = 2000)
    Y_new <- apply(X, 1, f0_x) + rnorm(2000)
    if (method == 1) {
        fit <- knn.reg(train = X, test = X_new, y = Y, k = k_value)
        Mean_EPE <- mean((fit$pred - Y_new)^2)/(2 * pi)
    }
    if (method == 2) {
        fit <- lm(Y ~ X)
        Y_hat <- X_new %*% fit$coefficients[2:6] + fit$coefficients[1]
        Mean_EPE <- mean((Y_hat - Y_new)^2)/(2 * pi)
    }
    return(Mean_EPE)
}
# The E(EPE(X)) of knn with k = 1,5,10,20,30,40
Expected_EPE(method = 1, X, Y, k_value = 1)

## [1] 1.96

Expected_EPE(method = 1, X, Y, k_value = 5)

## [1] 1.158

Expected_EPE(method = 1, X, Y, k_value = 10)

## [1] 1.051

Expected_EPE(method = 1, X, Y, k_value = 20)

## [1] 0.9898

Expected_EPE(method = 1, X, Y, k_value = 30)

## [1] 0.9692

Expected_EPE(method = 1, X, Y, k_value = 40)

## [1] 0.9627

# The E(EPE(X)) of linear fit
Expected_EPE(method = 2, X, Y)

## [1] 1.091
```