MAT2040 Project 2

Author: Bokai XU (119010355)

❖ Part I

➢ Model 1: Autoregression model

The predicting function is

$$x(t) = \sum_{n=1}^{N} a_n x(t-n)$$

That means we need to predict one day's price using the prices of previous N days.

For example, if we set $N = 5$, to predict the price at $t = t_0$, we can express the formula as:

$$x(t_0) = a_1 x(t_0 - 1) + a_2 x(t_0 - 2) + a_3 x(t_0 - 3) + a_4 x(t_0 - 4) + a_5 x(t_0 - 5)$$

where a_1, a_2, a_3, a_4, a_5 are all constants for a fixed N.

Then, let $t_0 = 6,7,8, \dots ,200$, we can get a linear system:

$$\begin{gathered} x(6) = a_1 x(5) + a_2 x(4) + a_3 x(3) + a_4 x(2) + a_5 x(1) \\ x(7) = a_1 x(6) + a_2 x(5) + a_3 x(4) + a_4 x(3) + a_5 x(2) \\ x(8) = a_1 x(7) + a_2 x(6) + a_3 x(5) + a_4 x(4) + a_5 x(3) \\ \dots \\ x(200) = a_1 x(199) + a_2 x(198) + a_3 x(197) + a_4 x(196) + a_5 x(195) \end{gathered}$$

Express the linear system into matrix form:

$$\begin{bmatrix} x(6) \\ x(7) \\ \vdots \\ x(199) \\ x(200) \end{bmatrix} = \begin{bmatrix} x(5) & x(4) & x(3) & x(2) & x(1) \\ x(6) & x(5) & x(4) & x(3) & x(2) \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ x(198) & x(197) & x(196) & x(195) & x(194) \\ x(199) & x(198) & x(197) & x(196) & x(195) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}$$

Let $A = \begin{bmatrix} x(5) & x(4) & x(3) & x(2) & x(1) \\ x(6) & x(5) & x(4) & x(3) & x(2) \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ x(198) & x(197) & x(196) & x(195) & x(194) \\ x(199) & x(198) & x(197) & x(196) & x(195) \end{bmatrix}$,

Use least square method to solve the linear system, that is,

$$b = Ax$$

$$\Longrightarrow A^T b = A^T A\hat{x}$$
$$\Longrightarrow (A^T A)^{-1} A^T b = \hat{x}$$

Use R function *lm()* to solve the problem.

(a) Data Preprocess

Use *read.table()* function to read a sequence of price data.
The raw data is a time series, which can be expressed by:

$$[x(1) \quad x(2) \quad \dots \quad x(199) \quad x(200)]$$

Then, convert the sequence data (raw data) into a data matrix with the form $D = \begin{bmatrix} x(6) & \\ x(7) & \\ \vdots & A \\ x(199) & \\ x(200) & \end{bmatrix} = [X1\ X2{\sim}X6]$.

```
all = c()
for (i in (n+1):length(price)){
  this = c()
  for (j in n:1){
    this = c(this, price[i-j])
  }
  this = c(price[i], this)
  all = c(all, this)
}
matrix_train = matrix(all, ncol = n+1, byrow = TRUE)
matrix_train = data.frame(matrix_train)
```

We convert both training data and testing data into this form.
The training data is saved as *matrix_train*, and the testing data is saved as *matrix_test*.

(b) Model building

We use *lm()* function to build a linear model:

```
result = lm(X1~., data = matrix_train)
```

(c) Predicting and Testing

With the model built, we can use this model to make prediction on testing dataset and get a predicted vector of prices:

```
pre_val = predict(result, matrix_test)
```

We also know actual values of price for our testing data:

```
act_val = matrix_test$X1
```

We use a criterion MSE on testing dataset:

$$MSE = \frac{\sum_{i=1}^{n}(predicted_i - actual_i)^2}{n}$$

We can compute the MSE using:

```
mse = sum((pre_val-act_val)^2)/length(pre_val)
```

(d) Model selecting

We can extend the above process to any N, so we encapsule this procedure to a function:

```
build <- function(n){

  all = c()
  for (i in (n+1):length(price)){
    this = c()
    for (j in n:1){
      this = c(this, price[i-j])
    }
    this = c(price[i], this)
    all = c(all, this)
  }
  matrix_train = matrix(all, ncol = n+1, byrow = TRUE)
  matrix_train = data.frame(matrix_train)
  result = lm(X1~., data = matrix_train)
  return(result)
}


test <- function(result, n){

  all = c()
```

```
  for (i in (n+1):length(test_price)){
    this = c()
    for (j in n:1){
      this = c(this, test_price[i-j])
    }
    this = c(test_price[i], this)

    all = c(all, this)
  }
  matrix_test = matrix(all, ncol = n+1, byrow = TRUE)
  matrix_test = data.frame(matrix_test)
  pre_val = predict(result, matrix_test)
  act_val = matrix_test$X1
  mse = sum((pre_val-act_val)^2)/length(pre_val)
  return(mse)
}
```

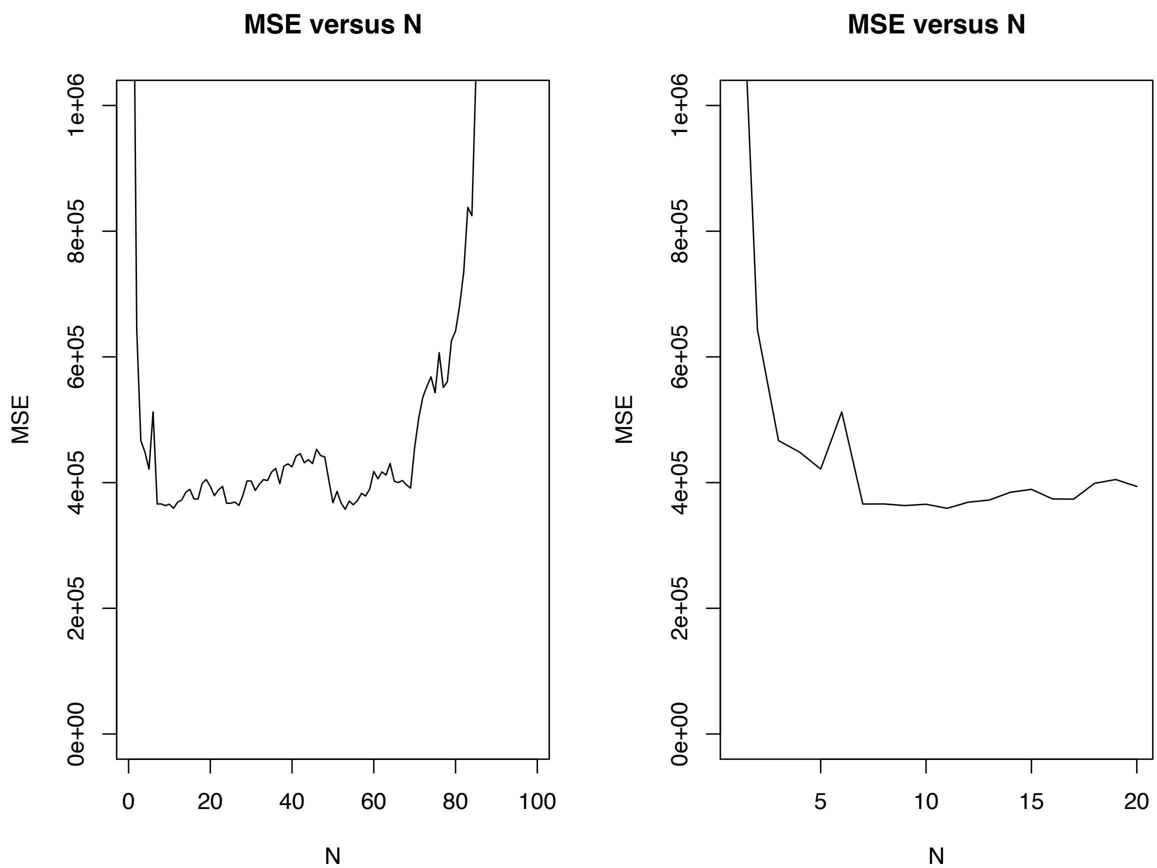
We choose some candidate N, and test their validity.

Here we choose $N = \{1,2,3, \dots ,99\}$, calculate their MSE.

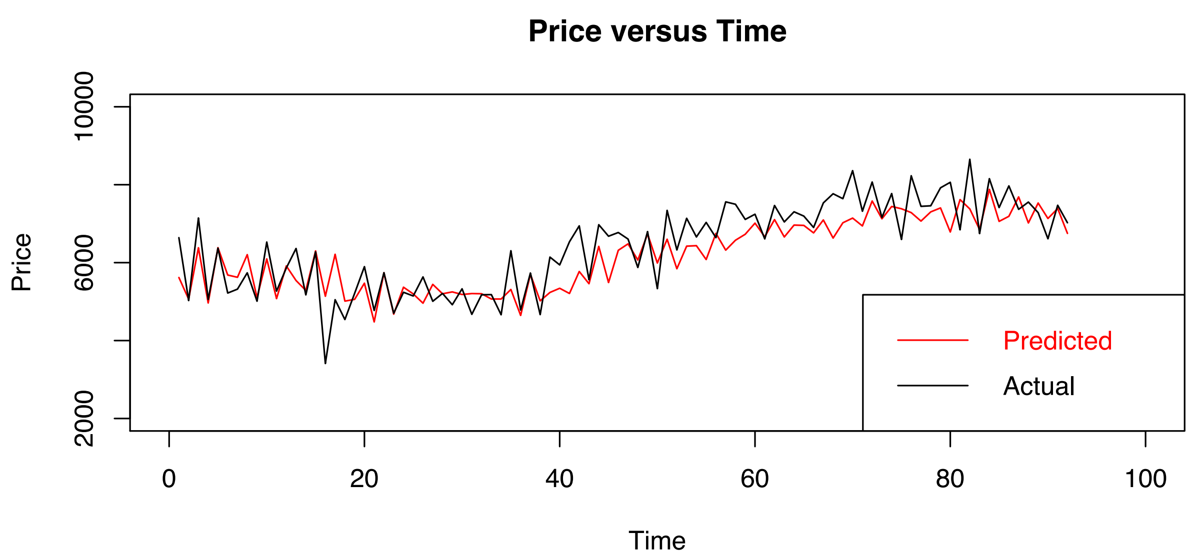
```
for (n in 1:99){
  result = build(n)
  mse = test(result, n)
}
```

The relationship between MSE and N is shown in the following plot. We notice that, the value decreases sharply as N increases, at $N = 8$, the value is minimized in short term. The value of MSE remains in this interval until $N = 80$. Because the validity of model $N = 8$ to model $N = 80$ is almost the same, we definitely choose a smaller and simpler model $N = 8$.

The optimal parameter N is $N = 8$ at which the $MSE = 366010.6$.



Then we pick the optimal model:



The predicting function is: (PP8 means the price 8 days ago, PP7 means the price 7 days ago, …)

```
Call:
lm(formula = X1 ~ ., data = matrix_train)

Residuals:
    Min      1Q   Median      3Q     Max
-1526.05  -372.48   -18.59   372.88  1326.35

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 749.440772 372.109223   2.014   0.0455 *
PP8           0.062789   0.055905   1.123   0.2628
PP7           0.116253   0.056401   2.061   0.0407 *
PP6          -0.062426   0.058533  -1.066   0.2876
PP5           0.341690   0.054689   6.248 2.84e-09 ***
PP4          -0.004616   0.055958  -0.082   0.9344
PP3           0.087143   0.069578   1.252   0.2120
PP2           0.423444   0.065987   6.417 1.16e-09 ***
PP1          -0.105990   0.073662  -1.439   0.1519
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 499.4 on 183 degrees of freedom
Multiple R-squared:  0.6033,  Adjusted R-squared:  0.5859
F-statistic: 34.78 on 8 and 183 DF,  p-value: < 2.2e-16
```

- Model 2: Fourier series model

Fourier series:

$$f(z) = \sum_{n=1}^{N} a_n \sin(nz) + b_n \cos(nz)$$

For this model, we need to first determine the interval for z, that is, Δz, and N.
Denote the order of data point as $o(i)$, that is, for the first data point, $o(1) = 1$, for the 200th data point, $o(200) = 200$.
Let $z = o(i) * \Delta z$. Now we use Fourier series to approximate the price function.

(a) Data preprocessing

Use *read.table()* function to read a sequence of price data.
The raw data is a time series, which can be expressed by:

$$[x(1) \quad x(2) \quad \dots \quad x(199) \quad x(200)]$$

Because there are 200 data point in the training data in the order of time, that means, we assign $o(1) = 1, o(2) = 2, o(3) = 3, \dots, o(200) = 200$. Then we compute the absolute time for each data point. $z(1) = 1 * \Delta z, z(2) = 2 * \Delta z, z(3) = 3 * \Delta z, \dots, z(200) = 200 * \Delta z$. With t known, we can immediately build a model for price with respect to t.

Now given an $N = 3$, we want to build a model such that:

$$f(x) \sim \sin(z) + \cos(z) + \sin(2z) + \cos(2z) + \sin(3z) + \cos(3z)$$

To make it a linear regression, we should first compute the values of $\sin(z), \cos(z), \sin(2z), \cos(2z), \sin(3z), \cos(3z)$ for each data point, and then perform linear regression.

Convert raw data into data frame in order to do regression:

```
y=price
n <- length(y)
tmp <- rep(NA, 2*pr*n)
col_p <- matrix(tmp, n, 2*pr)

for(k in 1:pr) {
  for(j in 1:n){
    col_p[j,k] <- cos(j*k/scale)
    col_p[j, pr+k] <- sin(j*k/scale)
  }
}
col_p <- data.frame(col_p)
```

(b) Model building and selecting

Assume the time interval $\Delta t = 5$ and $N = 3$, with computation, we get a data matrix similar to

$$D = \begin{bmatrix} x(1) & \sin(5) & \sin(10) & \sin(15) \\ x(2) & \sin(10) & \sin(20) & \sin(30) \\ \vdots & \vdots & \vdots & \vdots \\ x(200) & \sin(1*5*200) & \sin(2*5*200) & \sin(3*5*200) \end{bmatrix},$$

where $A = \begin{bmatrix} \sin(5) & \sin(10) & \sin(15) \\ \sin(10) & \sin(20) & \sin(30) \\ \vdots & \vdots & \vdots \\ \sin(1*5*200) & \sin(2*5*200) & \sin(3*5*200) \end{bmatrix},$

and

$$b = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(200) \end{bmatrix}$$

The objective is to find the least square solution:

$$(A^T A)^{-1} A^T b = \hat{x}$$

We will first determine the optimal value of time interval Δz, and pick an optimal N after determining Δz.

```
fit <- lm(y ~ ., data = col_p)
```

(c) Model testing

Here we use the same procedure as the way we treat training data:

```
for(k in 1:pr) {
  for(j in 1:n){
    col_p[j,k] <- cos((j+200)*k/scale)
    col_p[j, pr+k] <- sin((j+200)*k/scale)
  }
}
col_p <- data.frame(col_p)
```

Then we compute the corresponding MSE:

```
pred = predict(fit, col_p)
mse = sum((pred - test_price)^2)/length(pred)
```

We encapsule the above process into a function *obj3():*

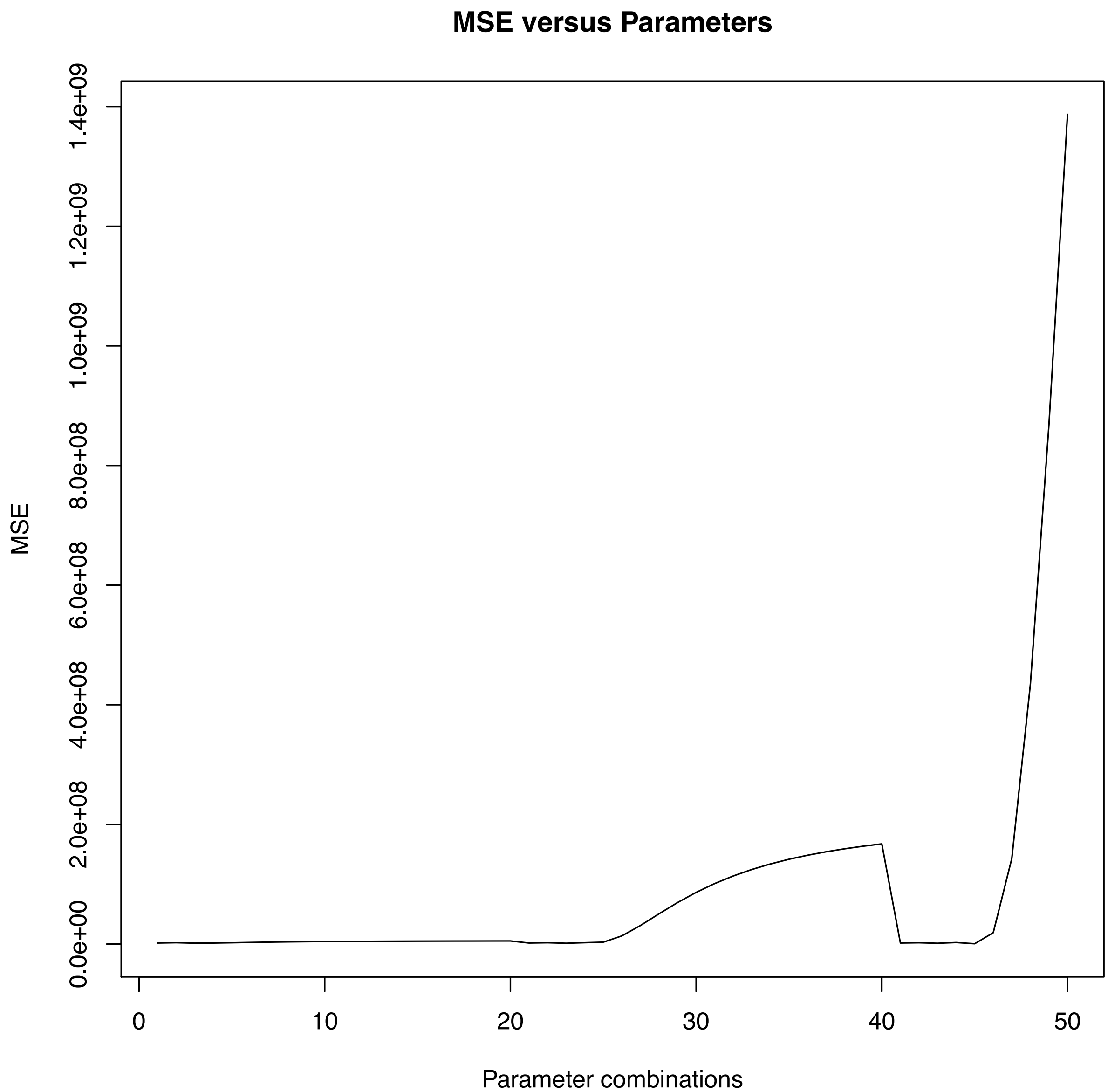
```
obj3 <- function(pr, scale){

  y=price
  n <- length(y)
  tmp <- rep(NA, 2*pr*n)
  col_p <- matrix(tmp, n, 2*pr)

  for(k in 1:pr) {
```

```
    for(j in 1:n){
      col_p[j,k] <- cos(j*k/scale)
      col_p[j, pr+k] <- sin(j*k/scale)
    }
  }

  col_p <- data.frame(col_p)
  fit <- lm(y ~ ., data = col_p)
  # summary(fit)
  y = test_price
  n <- length(y)
  tmp <- rep(NA, 2*pr*n)
  col_p <- matrix(tmp, n, 2*pr)

  for(k in 1:pr) {
    for(j in 1:n){
      col_p[j,k] <- cos((j+200)*k/scale)
      col_p[j, pr+k] <- sin((j+200)*k/scale)
    }
  }
  col_p <- data.frame(col_p)
  pred = predict(fit, col_p)
  mse = sum((pred - test_price)^2)/length(pred)
  return(mse)
}
```

(d) Model selecting

```
v = c()
for (pr in 1:20){
  for (scale in seq(10,200,by = 10)){
    this = obj3(pr, scale)
    v = c(v, this)
  }
}
```

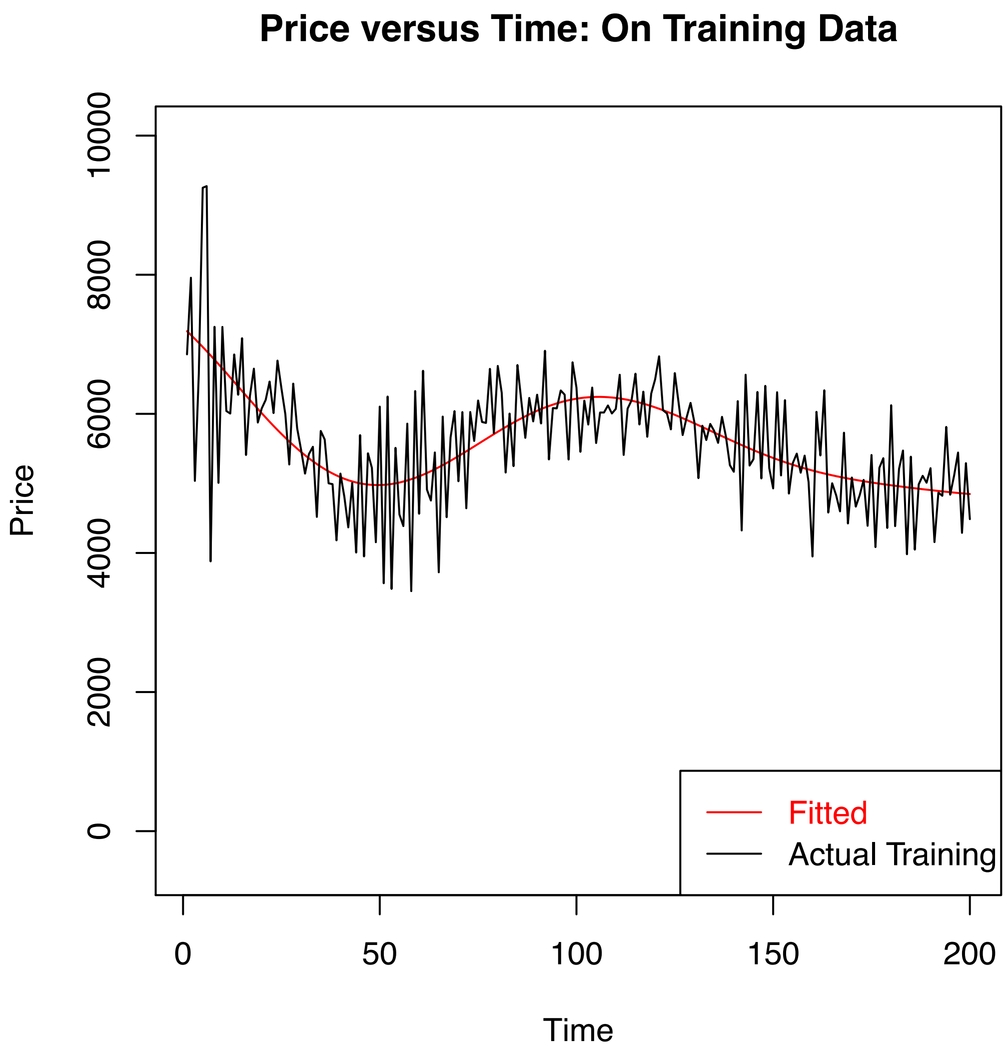
The optimization process is:



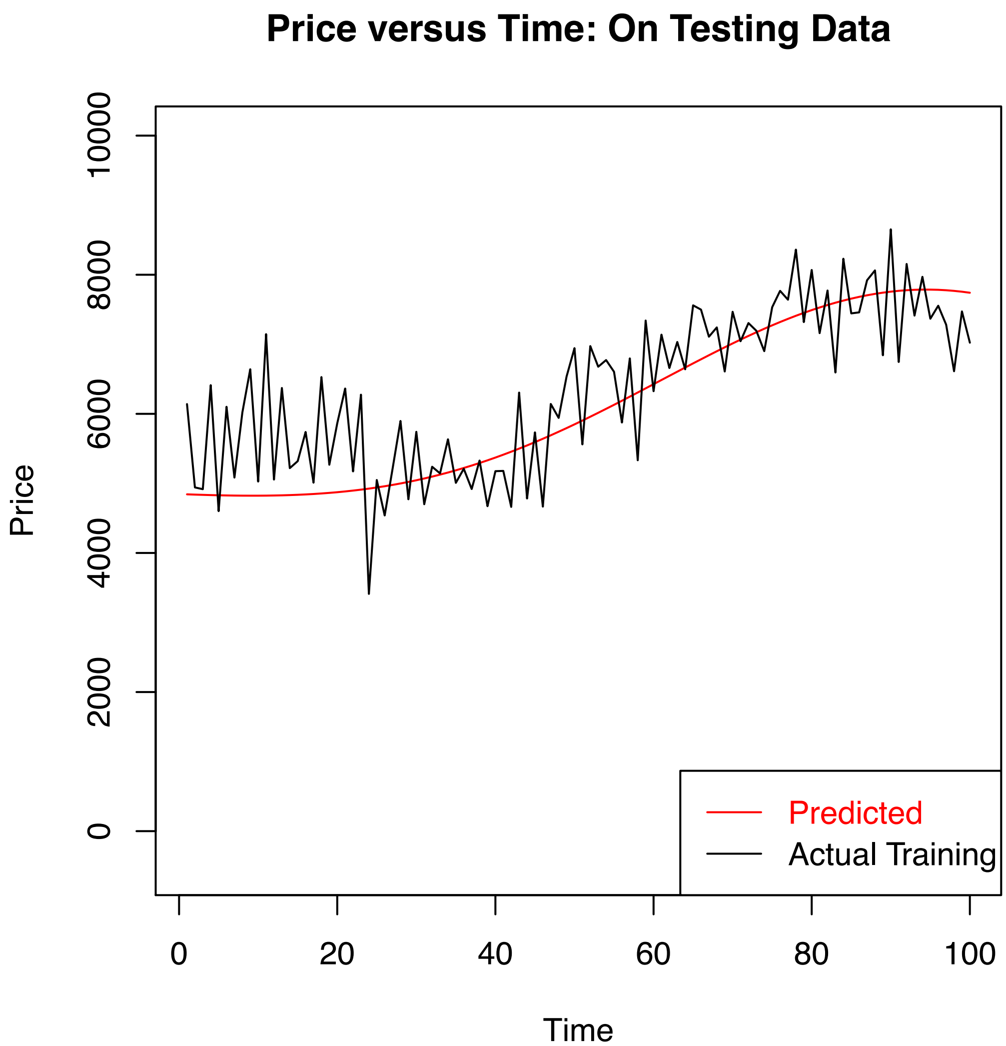
We pick the parameter combination with the lowest MSE.

Now we find that the optimal $N = 1$ and $\Delta z = 50$ (the 45th combination).

The following plot shows the fitting effect (plotting fitted curve and training data together):



Then we use the first 100 fitted values for training data (time from 1 to 100) **as** a prediction of testing data (testing data has 100 observations, time from 1 to 100).



Compute the MSE for this model:

The MSE is 534794.

The details of this model are:

```
Call:
lm(formula = y ~ ., data = col_p)

Residuals:
    Min      1Q  Median      3Q     Max
-2961.22 -370.60   18.77  401.32 2372.54

Coefficients:
```

```
           Estimate Std. Error t value Pr(>|t|)
(Intercept)  5843.30     398.18  14.675  <2e-16 ***
X1           745.58     312.70   2.384   0.0181 *
X2           392.91     316.08   1.243   0.2153
X3           259.21     211.00   1.228   0.2208
X4           -36.96     633.84  -0.058   0.9536
X5          -856.80     368.78  -2.323   0.0212 *
X6          -278.63     126.96  -2.195   0.0294 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 696.6 on 193 degrees of freedom
Multiple R-squared:  0.4027,  Adjusted R-squared:  0.3842
F-statistic: 21.69 on 6 and 193 DF,  p-value: < 2.2e-16
```

- Model 3: Polynomial Model

This model assumes the price function to be a sum of N polynomial variables:

$$f(z) = \sum_{n=1}^{N} a_n z^{n-1}$$

For this model, we need to first assign a time interval for z, that is, Δz, and N.

(a) Data preprocessing

Because there are 200 data point in the training data in the order of time, that means, we assign $o(1) = 1, o(2) = 2, o(3) = 3, \dots, o(200) = 200$. Then we compute the absolute time for each data point. $z(1) = 1 * \Delta z, z(2) = 2 * \Delta z, z(3) = 3 * \Delta z, \dots, z(200) = 200 * \Delta z$. With z known, we can immediately build a model for price with respect to z.

Now given an $N = 3$, we want to build a model such that:

$$f(z) \sim 1 + \mathrm{z} + \mathrm{z}^2 + z^3 + \cdots + z^{N-1}$$

To make it a linear regression, we should first compute the values of $1, \mathrm{z}, \mathrm{z}^2, z^3, \dots, z^{N-1}$ for each data point, and then perform linear regression.

We use a function to compute these values with given order $o(i)$, N, time interval Δz:

```
generate2 <- function(t,n,intv){
  this = c()
```

```
 for (i in 2:n){
   slice = (t*intv)^(i-1)
   this = c(this, slice)
 }

 return(this)
}
```

For a given sequence of price, for each data point, we perform the *generate()* function, compute their corresponding z^{n} values.

```
all = c()

for (i in 1:length(price)){
  ans = generate2(i,n,intv)
  ans = c(price[i], ans)
  all = c(all, ans)
}

matrix_train = matrix(all, ncol = n, byrow = TRUE)
matrix_train = data.frame(matrix_train)
```

(b) Model building

Assume the time interval $\Delta z = 5$ and $N = 3$, with computation, we get a data matrix similar to

$$D = \begin{bmatrix} x(1) & 1 & 5 & 5^2 \\ x(2) & 1 & 10 & 10^2 \\ \vdots & \vdots & \vdots & \vdots \\ x(200) & 1 & (5*200) & (5*200)^2 \end{bmatrix},$$

where $A = \begin{bmatrix} 1 & 5 & 5^2 \\ 1 & 10 & 10^2 \\ \vdots & \vdots & \vdots \\ 1 & (5*200) & (5*200)^2 \end{bmatrix},$

and $b = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(200) \end{bmatrix}$

The objective is to find the least square solution:

$$(A^T A)^{-1} A^T b = \hat{x}$$

Use *lm()* function to fit the model:

```
result = lm(X1~.,data = matrix_train)
```

(c) Predicting and Testing

To prepare the test data, we use the same procedure as what we do to training data.

```
all = c()

for (i in 1:length(test_price)){
  ans = generate2(i,n,intv)
  ans = c(test_price[i], ans)
  all = c(all, ans)
}

matrix_test = matrix(all, ncol = n, byrow = TRUE)
matrix_test = data.frame(matrix_test)
```

Use *predict()* function to give prediction values and then compute MSE.

```
pre_val = predict(result, matrix_test)
act_val = matrix_test$X1
mse = sum((pre_val-act_val)^2/length(test_price))
```

(d) Model selecting

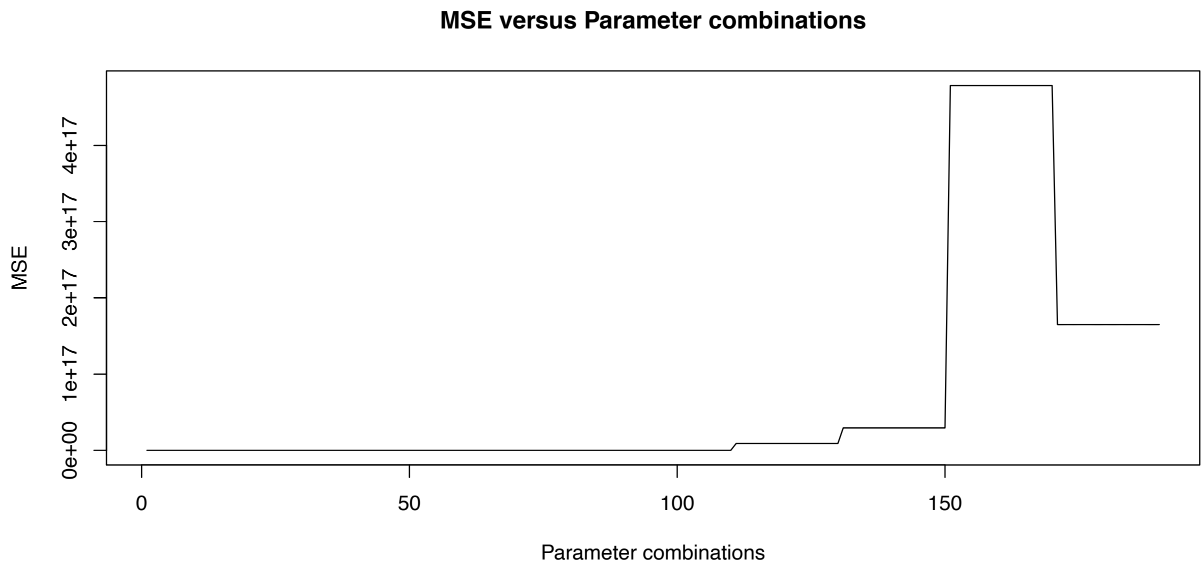
We encapsule the above process into a function *obj()*:

```
obj2 <- function(x){
 n = x[1]
 n = round(n,digits = 0)
 intv = x[2]
 all = c()

 for (i in 1:length(price)){
  ans = generate2(i,n,intv)
  ans = c(price[i], ans)
  all = c(all, ans)
 }
```

```
  matrix_train = matrix(all, ncol = n, byrow = TRUE)
  matrix_train = data.frame(matrix_train)

  result = lm(X1~.,data = matrix_train)

  all = c()

  for (i in 1:length(test_price)){
    ans = generate2(i,n,intv)
    ans = c(test_price[i], ans)
    all = c(all, ans)
  }

  matrix_test = matrix(all, ncol = n, byrow = TRUE)
  matrix_test = data.frame(matrix_test)

  pre_val = predict(result, matrix_test)
  act_val = matrix_test$X1

  mse = sum((pre_val-act_val)^2/length(test_price))
  return(mse)
}
```

Now, for each N in $\{2,3, \dots ,20\}$, we test Δz in $\{1,2,3, \dots ,10\}$. Then we have $19 * 10 = 190$ combinations of parameters.

Pick different combinations of parameters and compute MSE, we can obtain the pattern:
(The order is $\{N = 2, \Delta z = 1\}, \{N = 2, \Delta z = 2\}, \dots , \{N = 10, \Delta z = 10\}$)



We can find that the effect of Δz is 0, because when we deal with polynomials,

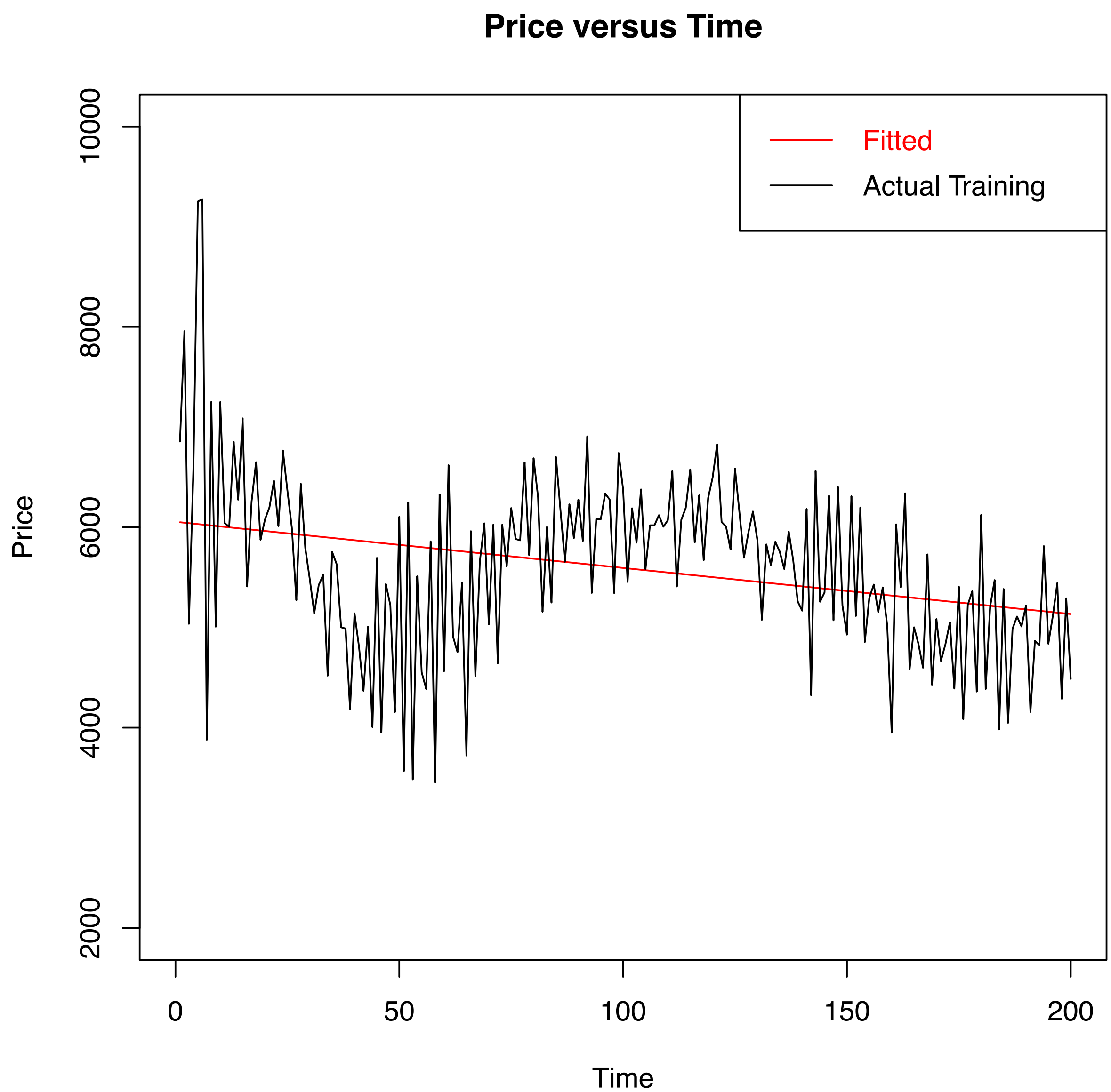
$$\beta_k(\alpha x)^k = \beta_k \alpha^k x^k = \beta_k(\alpha x)^k = \beta_k' x^k$$

Multiplying a factor on x does not affect the estimated value of coefficients.

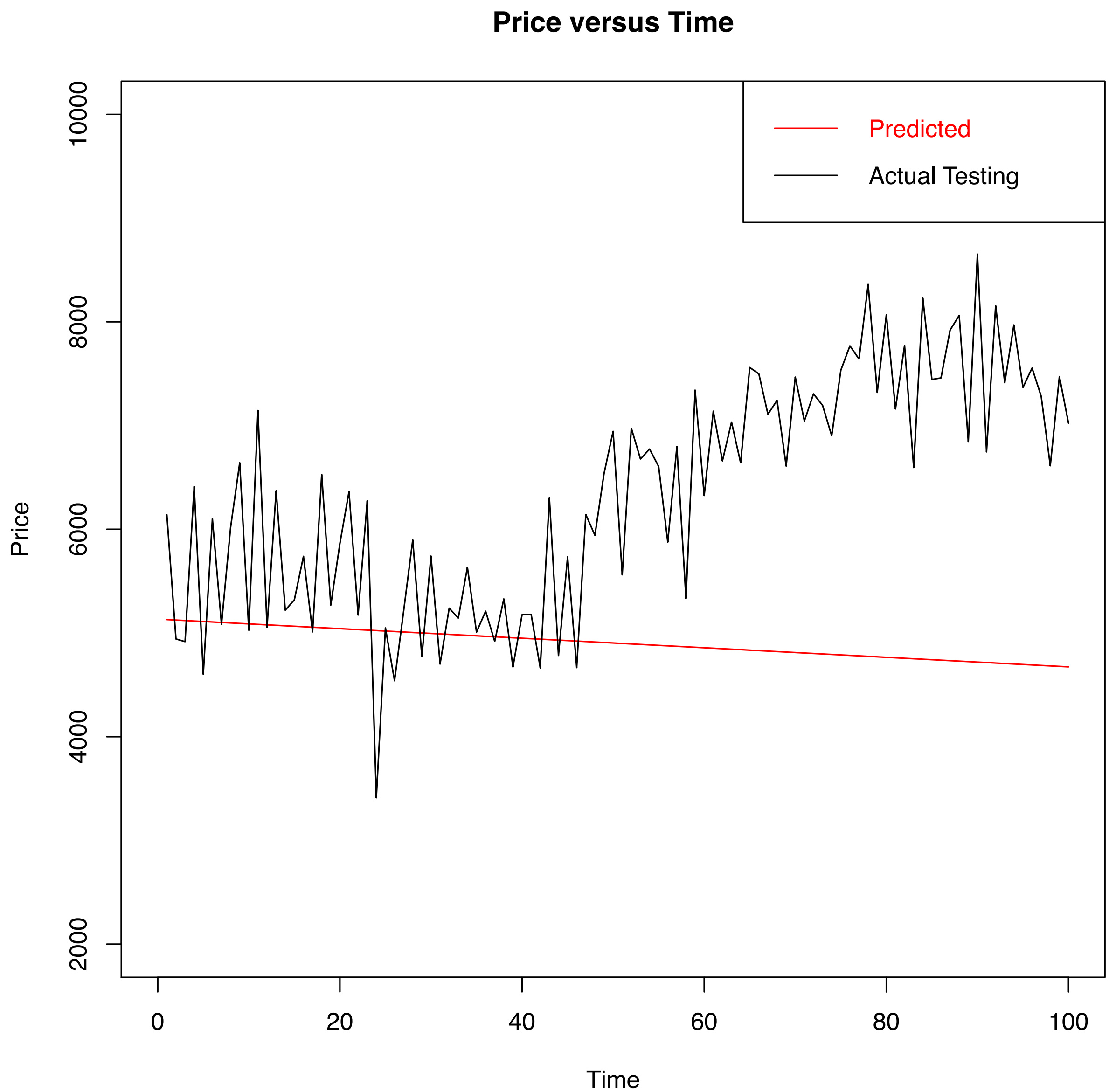
The MSE of this model only depends on N.

The optimal model is $N = 2$ where $MSE = 898040$.

The fitted curve on the training data is:



The prediction for testing data is:



MSE of this model is 1615626.

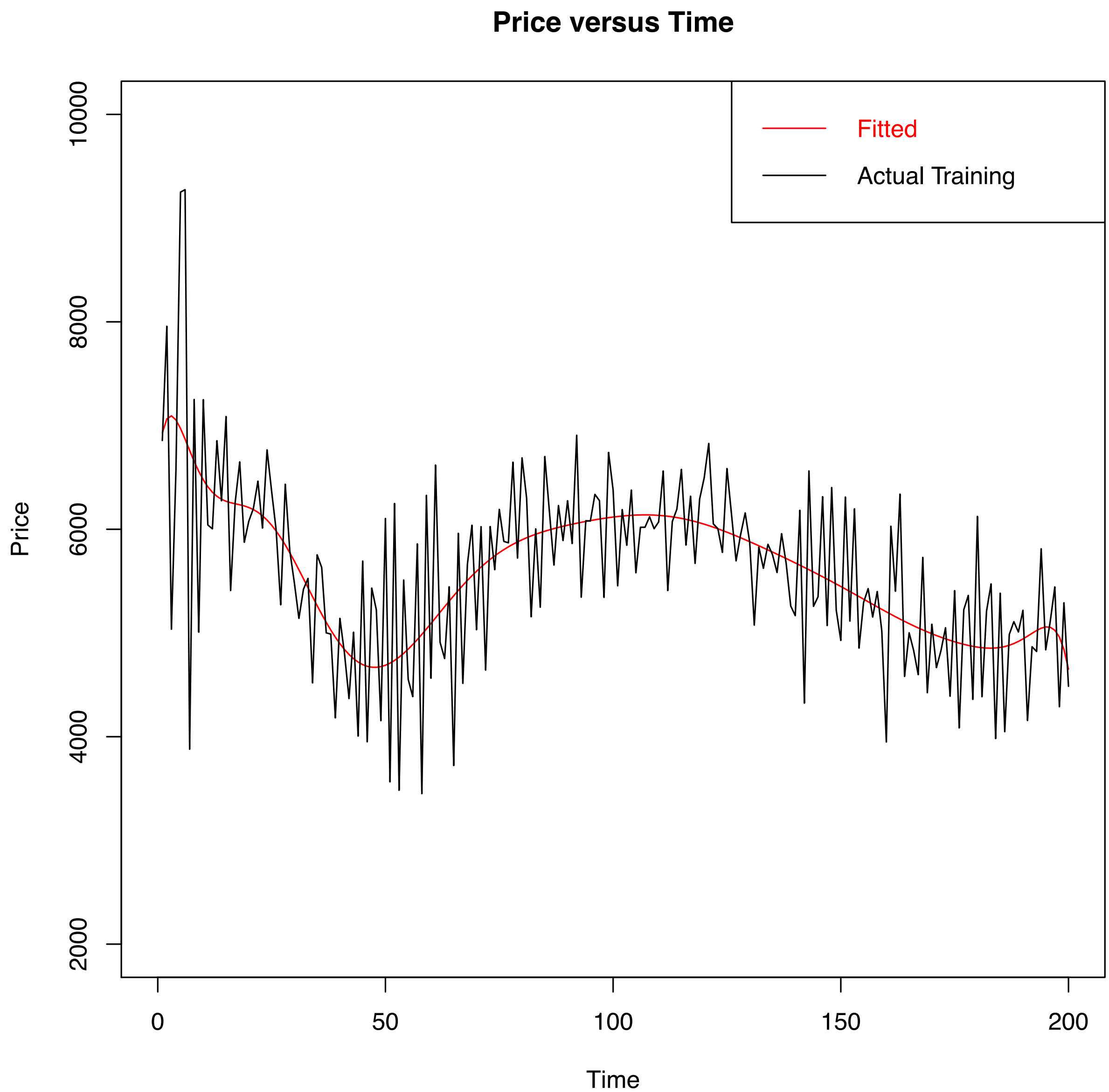
The details for this model:

```
Call:
lm(formula = X1 ~ ., data = matrix_train)

Residuals:
    Min      1Q  Median      3Q     Max
-2372.3  -440.6   105.2   486.0  3379.4

Coefficients:
```

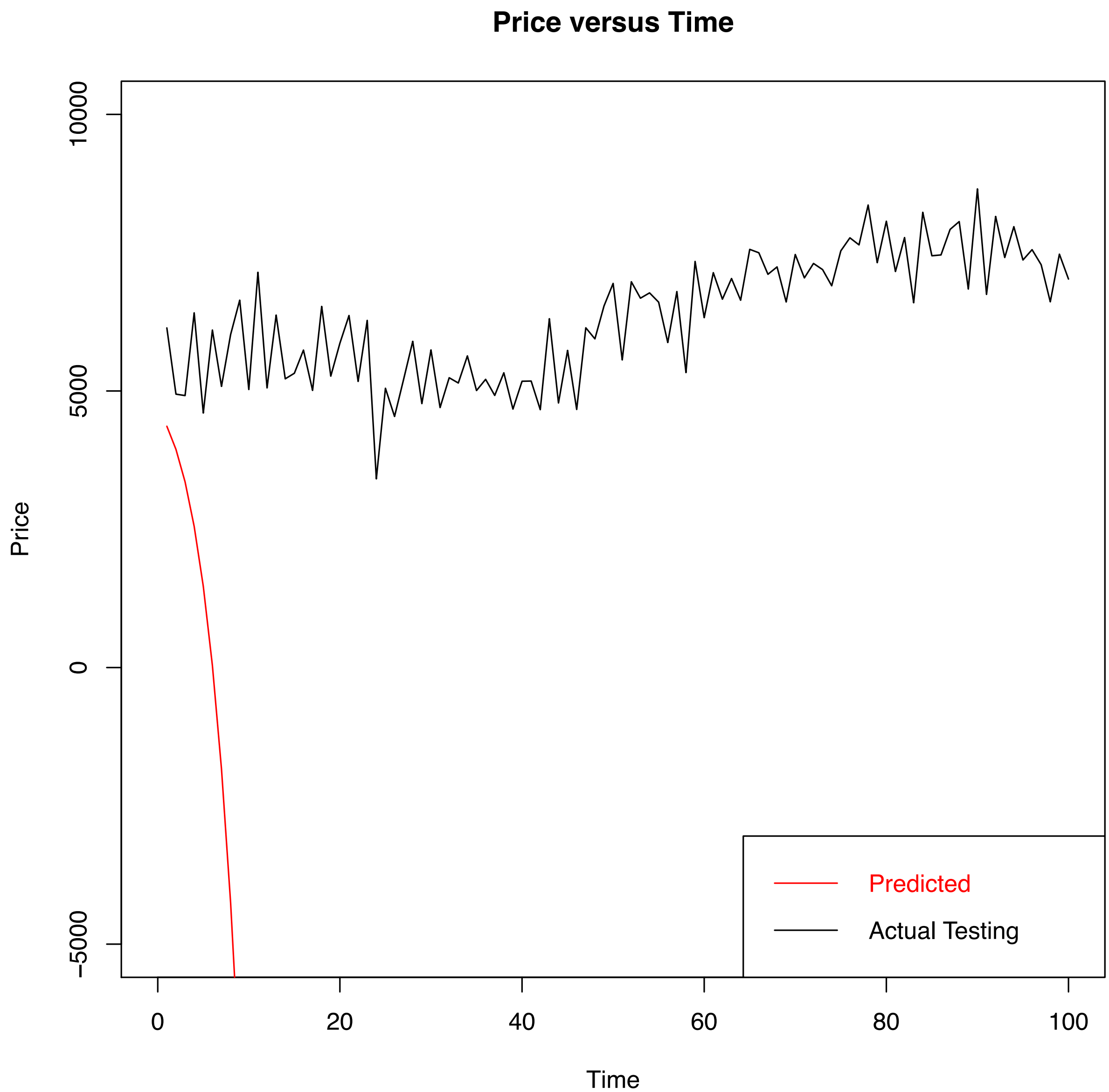
```
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 5894.67640  181.71781  32.439   <2e-16 ***
X2            0.13750    4.17442   0.033    0.974
X3           -0.02358    0.02012  -1.172    0.242
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 848.1 on 197 degrees of freedom
Multiple R-squared:  0.09638,Adjusted R-squared:  0.08721
F-statistic: 10.51 on 2 and 197 DF,  p-value: 4.619e-05
```

- Overfitting

Now we would like to discuss the overfitting phenomenon in this model:
We observed that a large N can lead to better fit for the training data set. For example, we assign $N = 18$.



The fitted curve matches the original training data very well. However, this curve will lead to a larger MSE on testing data set.



The MSE for $N = 18$ is 478551030806886336, which is much bigger than that of $N = 2$.

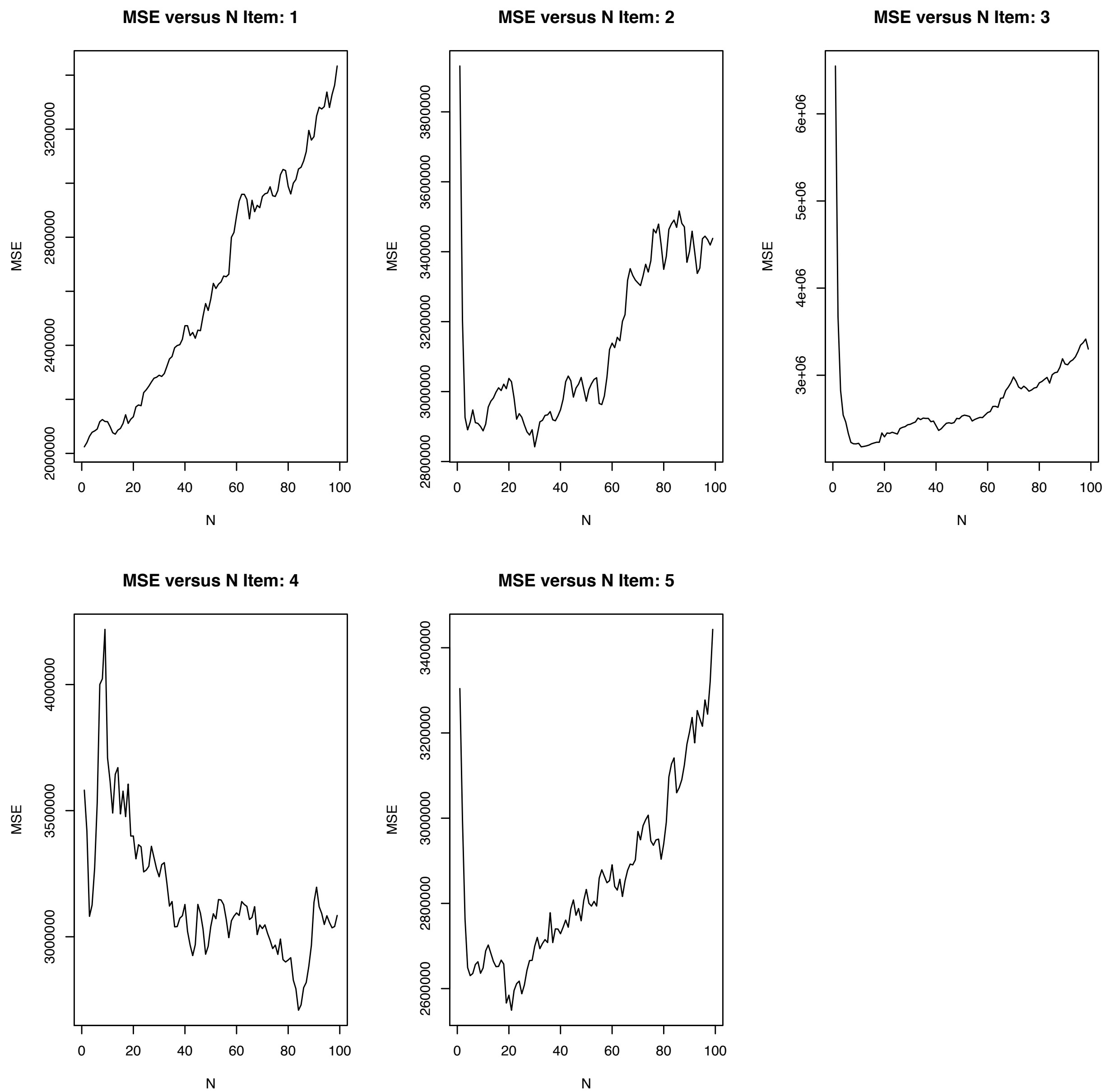
The details of this model are:

❖ Conclusion

The autoregression model get the lowest MSE on the testing data, at the value of around 360000. It is the best model.

- Part II
  - Without considering interaction

If there is no interaction between different commodities, we can use Model 1 from part 1 to build models. The result can be shown here:



- Considering the interaction

We can imagine the case that other commodities' prices may influence one commodity's price. If we denote today's prices (spot prices) for A, B, C, D, E be p_A, p_B, p_C, p_D, p_E we have some relationship:

$$\begin{aligned} p_A &= f(p_B, p_C, p_D, p_E) \\ p_B &= f(p_A, p_C, p_D, p_E) \\ p_C &= f(p_A, p_B, p_D, p_E) \\ p_D &= f(p_A, p_B, p_C, p_E) \\ p_E &= f(p_A, p_B, p_C, p_D) \end{aligned}$$

However, the five spot prices p_A, p_B, p_C, p_D, p_E are all unknowns. What we know is the previous prices data. If we want to know today's price of p_A, we need to know the price of the remaining four commodities. That is a dilemma.

Even if we made it to estimate the parameters of price function $f(\omega)$, it is impossible to know all 5 prices without knowing any one of these 5 spot prices.

However, we can make a hypothesis, that is, estimated value of spot price may be a function of the previous price of that commodity, that is,

$$\begin{aligned} \hat{p}_{A(Single)_t} &= f(p_{A_{t-1}}, p_{A_{t-2}}, \dots, p_{A_{t-N}}) \\ \hat{p}_{B(Single)_t} &= f\left(p_{B_{t-1}}, p_{B_{t-2}}, \dots, p_{B_{t-N}}\right) \\ \hat{p}_{C(Single)_t} &= f\left(p_{C_{t-1}}, p_{C_{t-2}}, \dots, p_{C_{t-N}}\right) \\ \hat{p}_{D(Single)_t} &= f\left(p_{D_{t-1}}, p_{D_{t-2}}, \dots, p_{D_{t-N}}\right) \\ \hat{p}_{E(Single)_t} &= f(p_{E_{t-1}}, p_{E_{t-2}}, \dots, p_{E_{t-N}}) \end{aligned}$$

The above five functions can be estimated using Model 1 in part 1, the next objective is to use the 5 estimated prices to predict the spot price with interaction.

Because $\hat{p}_{A(Single)_t}$ is a function of $p_{A_{t-1}}, p_{A_{t-2}}, \dots, p_{A_{t-N}}$

Then the estimated value of p_A with consideration of interaction can be computed from:

$$\hat{p}_{A_t} = f(\hat{p}_{A(Single)_t}, \hat{p}_{B(Single)_t}, \hat{p}_{C(Single)_t}, \hat{p}_{D(Single)_t}, \hat{p}_{E(Single)_t})$$

Because

$$\begin{aligned} \hat{p}_{A(Single)_t} &= f(p_{A_{t-1}}, p_{A_{t-2}}, \dots, p_{A_{t-N}}) \\ \hat{p}_{B(Single)_t} &= f\left(p_{B_{t-1}}, p_{B_{t-2}}, \dots, p_{B_{t-N}}\right) \\ \hat{p}_{C(Single)_t} &= f\left(p_{C_{t-1}}, p_{C_{t-2}}, \dots, p_{C_{t-N}}\right) \\ \hat{p}_{D(Single)_t} &= f\left(p_{D_{t-1}}, p_{D_{t-2}}, \dots, p_{D_{t-N}}\right) \\ \hat{p}_{E(Single)_t} &= f(p_{E_{t-1}}, p_{E_{t-2}}, \dots, p_{E_{t-N}}) \end{aligned}$$

Then we express

$$\hat{p}_{A_t} = f(p_{A_{t-1}}, p_{A_{t-2}}, \dots, p_{A_{t-N}}; p_{B_{t-1}}, p_{B_{t-2}}, \dots, p_{B_{t-N}}; p_{C_{t-1}}, p_{C_{t-2}}, \dots, p_{C_{t-N}}; p_{D_{t-1}}, p_{D_{t-2}}, \dots, p_{D_{t-N}}; p_{E_{t-1}}, p_{E_{t-2}}, \dots, p_{E_{t-N}})$$

The objective is to estimate $\hat{p}_{A_t}$ using $N * 5$ previous data points.

(a) Data preprocessing

So, we use R to do data preprocessing:

```
# Process raw data, extract previous n price -> n independent variables
all = c()
for (i in (n+1):dim(data_matrix)[1]){
  this = c(data_matrix[(i-n):(i-1),1:5]) # In total n lines
  this = c(data_matrix[i,pid], this)
  # print(this)
  all = c(all, this)
}

# Convert into data frame
data_p = matrix(all, ncol = n*5+1, byrow = TRUE)
data_p = data.frame(data_p)
# View(data_p)
```

(b) Model building

Use *lm()* and *predict()* function to build the model.

```
# Build linear model
result = lm(X1~., data = data_p)

# summary(result)
return(result)
```

We encapsule the above process into a function.

```
build <- function(n, pid){
 # Define the number of N = n

 # Process raw data, extract previous n price -> n independent variables
 all = c()
```

```r
 for (i in (n+1):dim(data_matrix)[1]){
   this = c(data_matrix[(i-n):(i-1),1:5]) # In total n lines
   this = c(data_matrix[i,pid], this)
   # print(this)
   all = c(all, this)
 }

 # Convert into data frame
 data_p = matrix(all, ncol = n*5+1, byrow = TRUE)
 data_p = data.frame(data_p)
 # View(data_p)

 # Build linear model
 result = lm(X1~., data = data_p)

 # summary(result)
 return(result)
}
```

(c) Model Testing

We will build five models to predict the five spot prices with consideration of interaction.

```r
# Testing function module

test <- function(result, n, pid){
 # Define the number of N = n

 all = c()
 for (i in (n+1):dim(test_data_matrix)[1]){
   this = c(test_data_matrix[(i-n):(i-1),1:5]) # In total n lines
   this = c(test_data_matrix[i,pid], this)
   # print(this)
   all = c(all, this)
 }

 # Convert into data frame
 test_data_p = matrix(all, ncol = n*5+1, byrow = TRUE)
 test_data_p = data.frame(test_data_p)
 # View(test_data_p)

 # Use existing model to predict the testing data set
 pre_val = predict(result, test_data_p)
```

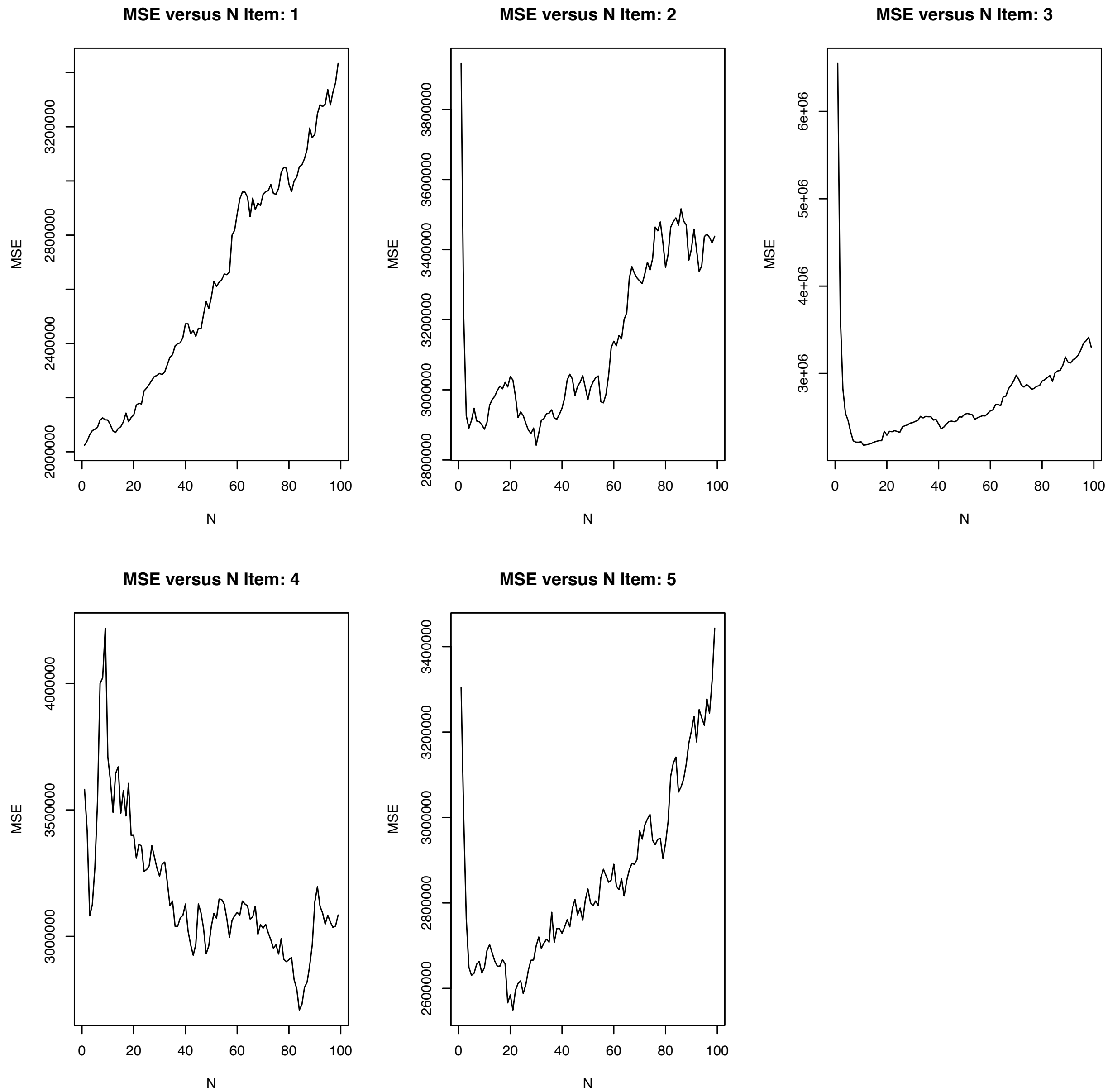
```
  # Actual value
  act_val = test_data_p$X1

  # Calculate MSE
  mse = sum((pre_val-act_val)^2)/length(pre_val)

  return(mse)
}
```

(d) Model selecting

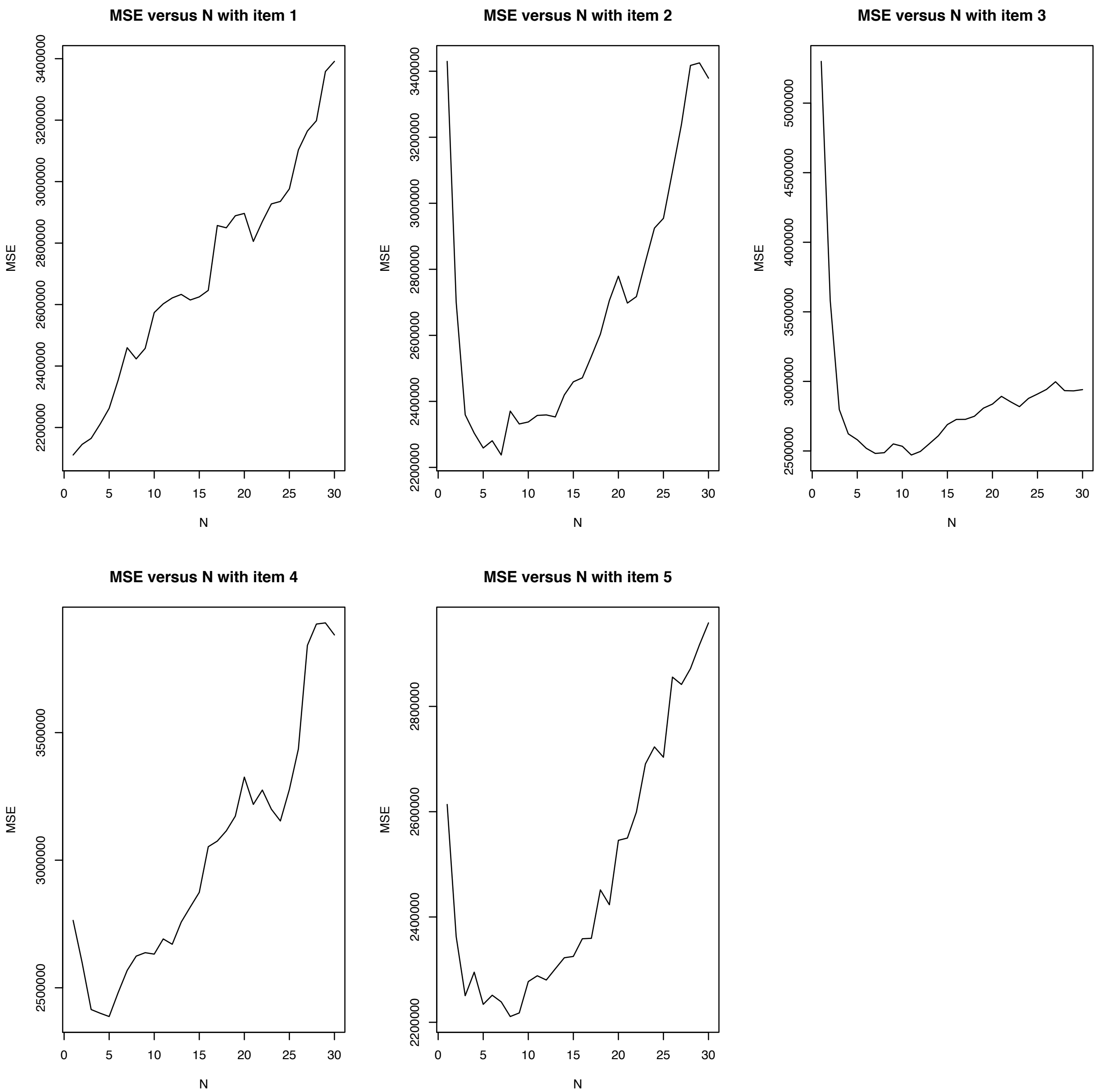
```
for (pid in 1:5){
  mse_v = c()

  for (n in 1:30){
  result = build(n, pid)
  mse = test(result, n, pid)
  mse_v = c(mse_v, mse)
  }

  y = mse_v
}
```

The result for 5 commodities is shown here:

- Without interaction



- With interaction



Compare the two scenarios, we can find:

(1) Commodity 1: The AR model is no use to predict the price of this commodity. The optimal MSE are all around 2200000, and optimal N is $N = 1$.
(2) Commodity 2: The interaction effect in considerable. Interaction does help to improve the prediction quality. The optimal MSE without consideration of interaction is around 2800000, however, the
(3) Commodity 3: The interaction effect seems to be not useful, in turn, it shrank the prediction correctness.

(4) Commodity 4: The interaction effect helps to decrease the optimal N, making the model simpler and interpretable. The original optimal N is around $N = 80$, which is computational complex and hard to interpret. Now, the optimal N becomes $N = 5$, which is easier to compute and easier to interpret. The optimal MSE also decreases. The original one is around 2600000. The new optimal MSE is around 2350000.

(5) Commodity 5: The interaction effect helps to decrease the optimal N, making the model simpler and interpretable. The original optimal N is around $N = 20$, which is computational complex and hard to interpret. Now, the optimal N becomes $N = 8$, which is easier to compute and easier to interpret. The optimal MSE also decreases. The original one is around 2500000. The new optimal MSE is around 2200000.

The final result is:

Commodity 1:

```
Call:
lm(formula = X1 ~ ., data = data_p)

Coefficients:
(Intercept)        X2         X3         X4         X5         X6
 189.211859   0.999887  -0.015316  -0.008035  -0.011690  -0.010666
```

```
Call:
lm(formula = X1 ~ ., data = data_p)

Residuals:
    Min      1Q  Median      3Q     Max
-4860.5  -847.1    70.1  1041.5  3228.1

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 189.211859 166.802370   1.134    0.257
X2            0.999887   0.014634  68.326   <2e-16 ***
X3           -0.015316   0.027956  -0.548    0.584
X4           -0.008035   0.028506  -0.282    0.778
X5           -0.011690   0.041541  -0.281    0.779
X6           -0.010666   0.035479  -0.301    0.764
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1402 on 493 degrees of freedom
Multiple R-squared:  0.9876,	Adjusted R-squared:  0.9874
F-statistic:  7829 on 5 and 493 DF,  p-value: < 2.2e-16
```

That suggests, the spot price of commodity 1 has strong relationship between the commodity 1's price on the previous day. It does not depend on the other commodities.

Commodity 2:

```
Call:
lm(formula = X1 ~ ., data = data_p)

Coefficients:
(Intercept)        X2         X3         X4         X5         X6         X7
 -1.585e+02  2.992e-02  5.010e-02  6.097e-02  9.143e-02  7.950e-02  -2.768e-02
        X8         X9        X10        X11        X12        X13        X14
 1.588e-01  3.342e-02  -5.124e-02  -2.835e-02  7.598e-02  5.252e-02  -2.011e-02
       X15        X16        X17        X18        X19        X20        X21
 -5.680e-02  1.660e-01  1.652e-01  1.488e-01  9.040e-02  1.206e-01  1.036e-01
       X22        X23        X24        X25        X26        X27        X28
 7.878e-02  2.070e-02  -5.132e-02  -1.692e-02  -4.494e-02  6.086e-02  2.990e-02
       X29        X30        X31        X32        X33        X34        X35
 -3.003e-02  4.688e-02  -2.215e-02  8.298e-04  4.486e-02  5.440e-02  -1.645e-02
       X36
 1.659e-02
```

```
Call:
lm(formula = X1 ~ ., data = data_p)

Residuals:
    Min      1Q  Median      3Q     Max
-3238.7  -936.7   -57.4   883.7  3594.1

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.585e+02  1.790e+02  -0.885 0.376400

X2           2.992e-02  5.321e-02   0.562 0.574254
X3           5.010e-02  6.728e-02   0.745 0.456823
X4           6.097e-02  6.740e-02   0.905 0.366121
X5           9.143e-02  6.744e-02   1.356 0.175864
X6           7.950e-02  6.820e-02   1.166 0.244351

X7          -2.768e-02  6.788e-02  -0.408 0.683597
X8           1.588e-01  4.697e-02   3.380 0.000787 ***
```

X9	*3.342e-02*	*3.773e-02*	*0.886*	*0.376183*	
X10	*-5.124e-02*	*4.034e-02*	*-1.270*	*0.204679*	
X11	*-2.835e-02*	*4.145e-02*	*-0.684*	*0.494307*	
X12	*7.598e-02*	*4.286e-02*	*1.773*	*0.076894*	*.*
X13	*5.252e-02*	*4.369e-02*	*1.202*	*0.229889*	
X14	*-2.011e-02*	*4.406e-02*	*-0.456*	*0.648319*	
X15	*-5.680e-02*	*4.488e-02*	*-1.265*	*0.206377*	
X16	*1.660e-01*	*5.021e-02*	*3.306*	*0.001022*	**
X17	*1.652e-01*	*4.914e-02*	*3.362*	*0.000838*	***
X18	*1.488e-01*	*4.887e-02*	*3.044*	*0.002465*	**
X19	*9.040e-02*	*4.894e-02*	*1.847*	*0.065387*	*.*
X20	*1.206e-01*	*4.871e-02*	*2.476*	*0.013634*	*
X21	*1.036e-01*	*4.807e-02*	*2.155*	*0.031712*	*
X22	*7.878e-02*	*4.763e-02*	*1.654*	*0.098793*	*.*
X23	*2.070e-02*	*4.458e-02*	*0.464*	*0.642665*	
X24	*-5.132e-02*	*4.473e-02*	*-1.147*	*0.251932*	
X25	*-1.692e-02*	*4.543e-02*	*-0.372*	*0.709694*	
X26	*-4.494e-02*	*4.558e-02*	*-0.986*	*0.324686*	
X27	*6.086e-02*	*4.523e-02*	*1.346*	*0.179076*	
X28	*2.990e-02*	*4.528e-02*	*0.660*	*0.509333*	
X29	*-3.003e-02*	*4.558e-02*	*-0.659*	*0.510292*	
X30	*4.688e-02*	*4.123e-02*	*1.137*	*0.256166*	
X31	*-2.215e-02*	*4.338e-02*	*-0.511*	*0.609853*	
X32	*8.298e-04*	*4.388e-02*	*0.019*	*0.984922*	
X33	*4.486e-02*	*4.399e-02*	*1.020*	*0.308405*	
X34	*5.440e-02*	*4.460e-02*	*1.220*	*0.223232*	
X35	*-1.645e-02*	*4.486e-02*	*-0.367*	*0.714018*	
X36	*1.659e-02*	*4.621e-02*	*0.359*	*0.719706*	

---

*Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1*

Residual standard error: 1408 on 457 degrees of freedom
Multiple R-squared: 0.9304, Adjusted R-squared: 0.9251
F-statistic: 174.7 on 35 and 457 DF, p-value: < 2.2e-16

That suggests spot price of commodity 2 depends on the price of commodity 2 of 6 days ago and of 4 days ago. And it depends on price of commodity 1 of 4 days ago. It also depends on the price of commodity 5 of 5 days ago. Their effects are all positive. That means, a higher price of

commodity 2 of 6 days ago and of 4 days ago
commodity 1 of 4 days ago
commodity 5 of 5 days ago

will denote a higher spot price of commodity 2.

Commodity 3:

```
Call:
lm(formula = X1 ~ ., data = data_p)

Coefficients:
(Intercept)         X2          X3          X4          X5          X6          X7
 1.279e+02   5.955e-02  -5.088e-02  -1.088e-02  -1.077e-01   6.486e-02   9.109e-02
        X8          X9         X10         X11         X12         X13         X14
-6.323e-02  -2.623e-02  -6.214e-03   1.985e-02  -7.256e-04   2.884e-02  -9.084e-03
       X15         X16         X17         X18         X19         X20         X21
 5.329e-02   1.177e-01   1.560e-01   2.875e-02   1.562e-01   1.837e-01   1.584e-01
       X22         X23         X24         X25         X26         X27         X28
 1.436e-01   2.971e-02   1.412e-02   6.355e-04  -7.960e-03   7.031e-02  -1.114e-01
       X29         X30         X31         X32         X33         X34         X35
 2.779e-02   3.723e-02  -6.782e-02   4.578e-02   2.622e-03  -5.897e-02  -2.912e-02
       X36
-1.547e-02
```

```
Call:
lm(formula = X1 ~ ., data = data_p)

Residuals:
   Min     1Q  Median     3Q    Max
-4142.9 -876.7   -17.0  870.7 3554.4

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.279e+02  1.742e+02   0.734 0.463146
X2           5.955e-02  5.176e-02   1.150 0.250597
X3          -5.088e-02  6.545e-02  -0.777 0.437269
X4          -1.088e-02  6.557e-02  -0.166 0.868229
X5          -1.077e-01  6.561e-02  -1.642 0.101363
X6           6.486e-02  6.634e-02   0.978 0.328744
```

```
X7         9.109e-02  6.604e-02   1.379 0.168452
X8        -6.323e-02  4.569e-02  -1.384 0.167077
X9        -2.623e-02  3.670e-02  -0.715 0.475105
X10       -6.214e-03  3.924e-02  -0.158 0.874252
X11        1.985e-02  4.032e-02   0.492 0.622802

X12       -7.256e-04  4.169e-02  -0.017 0.986121
X13        2.884e-02  4.250e-02   0.679 0.497702
X14       -9.083e-03  4.286e-02  -0.212 0.832271
X15        5.329e-02  4.366e-02   1.221 0.222904
X16        1.177e-01  4.884e-02   2.410 0.016343 *

X17        1.560e-01  4.780e-02   3.263 0.001185 **
X18        2.875e-02  4.754e-02   0.605 0.545646
X19        1.562e-01  4.761e-02   3.281 0.001113 **
X20        1.837e-01  4.738e-02   3.877 0.000121 ***
X21        1.584e-01  4.676e-02   3.387 0.000769 ***

X22        1.436e-01  4.633e-02   3.099 0.002061 **
X23        2.971e-02  4.336e-02   0.685 0.493681
X24        1.412e-02  4.352e-02   0.324 0.745790
X25        6.355e-04  4.420e-02   0.014 0.988534
X26       -7.961e-03  4.434e-02  -0.180 0.857614

X27        7.031e-02  4.400e-02   1.598 0.110694
X28       -1.114e-01  4.405e-02  -2.529 0.011783 *
X29        2.779e-02  4.434e-02   0.627 0.531179
X30        3.723e-02  4.011e-02   0.928 0.353818
X31       -6.782e-02  4.220e-02  -1.607 0.108740

X32        4.578e-02  4.269e-02   1.072 0.284093
X33        2.622e-03  4.280e-02   0.061 0.951174
X34       -5.897e-02  4.339e-02  -1.359 0.174775
X35       -2.912e-02  4.364e-02  -0.667 0.504832
X36       -1.547e-02  4.496e-02  -0.344 0.730939
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1369 on 457 degrees of freedom
Multiple R-squared:  0.8719,   Adjusted R-squared:  0.8621
F-statistic: 88.85 on 35 and 457 DF,  p-value: < 2.2e-16
```

That suggests spot price of commodity 3 depends on the price of commodity 1 of 3 days ago and of 4 days ago. And it depends on price of commodity 3 of 4 days ago. It also depends on the price of commodity 4 of 4 days ago and commodity 5 of 4 days ago. Their effects are all positive. That means, a higher price of

commodity 1 of 3 days ago and of 4 days ago
commodity 3 of 4 days ago
commodity 4 of 4 days ago
commodity 5 of 4 days ago

will denote a higher spot price of commodity 3.

Commodity 4:

```
Call:
lm(formula = X1 ~ ., data = data_p)

Coefficients:
(Intercept)        X2          X3          X4          X5          X6          X7
 251.517745    0.102058   -0.022279   -0.027981   -0.093031    0.045060    0.022249
        X8          X9         X10         X11         X12         X13         X14
 -0.065800    0.067985   -0.035530   -0.010849    0.096711    0.023114   -0.004644
       X15         X16         X17         X18         X19         X20         X21
 -0.005728   -0.031881   -0.019470    0.045726    0.079880   -0.001217    0.003219
       X22         X23         X24         X25         X26
 -0.020514   -0.002358    0.007169    0.114366    0.024650
```

```
Call:
lm(formula = X1 ~ ., data = data_p)

Residuals:
    Min      1Q  Median      3Q     Max
-3824.4  -908.3   -18.4   942.0  4727.0

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 251.517745 182.479424   1.378   0.1688
X2            0.102058   0.054212   1.883   0.0604 .
X3           -0.022279   0.068180  -0.327   0.7440
X4           -0.027981   0.068422  -0.409   0.6828
X5           -0.093031   0.068854  -1.351   0.1773
X6            0.045060   0.047879   0.941   0.3471

X7            0.022249   0.038746   0.574   0.5661
X8           -0.065800   0.040987  -1.605   0.1091
```

```
X9         0.067985  0.041787  1.627  0.1044
X10       -0.035530  0.043562 -0.816  0.4151
X11       -0.010849  0.044574 -0.243  0.8078

X12        0.096711  0.049462  1.955  0.0511 .
X13        0.023114  0.048650  0.475  0.6349
X14       -0.004644  0.047524 -0.098  0.9222
X15       -0.005728  0.047791 -0.120  0.9047
X16       -0.031881  0.048187 -0.662  0.5085

X17       -0.019470  0.045783 -0.425  0.6708
X18        0.045726  0.045913  0.996  0.3198
X19        0.079880  0.046054  1.735  0.0835 .
X20       -0.001217  0.045965 -0.026  0.9789
X21        0.003219  0.045787  0.070  0.9440

X22       -0.020514  0.041823 -0.490  0.6240
X23       -0.002358  0.043832 -0.054  0.9571
X24        0.007169  0.044395  0.161  0.8718
X25        0.114366  0.044770  2.554  0.0109 *
X26        0.024650  0.045407  0.543  0.5875
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1452 on 469 degrees of freedom
Multiple R-squared:  0.09329,Adjusted R-squared:  0.04496
F-statistic:  1.93 on 25 and 469 DF,  p-value: 0.004849
```

That suggests spot price of commodity 4 depends on the price of commodity 1 of 5 days ago and of 3 days ago. And it depends on price of commodity 4 of 1 day ago. Their effects are all positive. That means, a higher price of

commodity 1 of 5 days ago and of 3 days ago
commodity 4 of 1 days ago

will denote a higher spot price of commodity 4.

Commodity 5:

```
Call:
lm(formula = X1 ~ ., data = data_p)
```

```
Coefficients:
(Intercept)         X2          X3          X4          X5          X6          X7
 34.8552386  -0.0521900   0.1247272  -0.0748236  -0.0169437   0.0861467  -
0.0250207
       X8          X9         X10         X11         X12         X13         X14
 0.0442569  -0.0189617  -0.0015539  -0.0203260  -0.0138381   0.0648231   0.0320891
      X15         X16         X17         X18         X19         X20         X21
 0.0390481   0.0268725  -0.0006336  -0.0947193   0.0573919  -0.0790841   0.0040778
      X22         X23         X24         X25         X26         X27         X28
 0.0522329  -0.0718668  -0.0048793   0.0844191   0.0008355  -0.0482493   0.0583221
      X29         X30         X31         X32         X33         X34         X35
-0.0040846  -0.0621618   0.0772739   0.0349849   0.0924481  -0.0350924   0.0006968
      X36         X37         X38         X39         X40         X41
 0.0542138   0.0009964   0.0926442  -0.0143971  -0.0097024   0.0234902
```

```
Call:
lm(formula = X1 ~ ., data = data_p)

Residuals:
    Min      1Q  Median      3Q     Max
-3712.8  -929.0    34.3   924.4  4600.2

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.486e+01  1.789e+02   0.195   0.8456

X2          -5.219e-02  5.347e-02  -0.976   0.3296
X3           1.247e-01  6.699e-02   1.862   0.0633 .
X4          -7.482e-02  6.718e-02  -1.114   0.2660
X5          -1.694e-02  6.756e-02  -0.251   0.8021
X6           8.615e-02  6.793e-02   1.268   0.2054

X7          -2.502e-02  6.772e-02  -0.369   0.7119
X8           4.426e-02  6.755e-02   0.655   0.5127
X9          -1.896e-02  4.701e-02  -0.403   0.6869
X10         -1.554e-03  3.758e-02  -0.041   0.9670
X11         -2.033e-02  4.018e-02  -0.506   0.6132

X12         -1.384e-02  4.125e-02  -0.335   0.7374
X13          6.482e-02  4.281e-02   1.514   0.1307
X14          3.209e-02  4.368e-02   0.735   0.4630
X15          3.905e-02  4.399e-02   0.888   0.3752
X16          2.687e-02  4.475e-02   0.600   0.5485
```

```
X17     -6.336e-04 4.648e-02 -0.014  0.9891
X18     -9.472e-02 5.112e-02 -1.853  0.0646 .
X19      5.739e-02 5.042e-02  1.138  0.2557
X20     -7.908e-02 4.915e-02 -1.609  0.1083
X21      4.078e-03 4.942e-02  0.083  0.9343

X22      5.223e-02 4.961e-02  1.053  0.2930
X23     -7.187e-02 4.857e-02 -1.480  0.1397
X24     -4.879e-03 4.817e-02 -0.101  0.9194
X25      8.442e-02 4.773e-02  1.769  0.0776 .
X26      8.355e-04 4.434e-02  0.019  0.9850

X27     -4.825e-02 4.455e-02 -1.083  0.2794
X28      5.832e-02 4.517e-02  1.291  0.1973
X29     -4.085e-03 4.538e-02 -0.090  0.9283
X30     -6.216e-02 4.536e-02 -1.371  0.1712
X31      7.727e-02 4.540e-02  1.702  0.0894 .

X32      3.498e-02 4.546e-02  0.770  0.4419
X33      9.245e-02 4.690e-02  1.971  0.0493 *
X34     -3.509e-02 4.111e-02 -0.854  0.3938
X35      6.968e-04 4.324e-02  0.016  0.9872
X36      5.421e-02 4.374e-02  1.239  0.2159

X37      9.964e-04 4.433e-02  0.022  0.9821
X38      9.264e-02 4.450e-02  2.082  0.0379 *
X39     -1.440e-02 4.464e-02 -0.323  0.7472
X40     -9.702e-03 4.594e-02 -0.211  0.8328
X41      2.349e-02 4.693e-02  0.501  0.6169
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1396 on 451 degrees of freedom
Multiple R-squared:  0.6207,  Adjusted R-squared:  0.587
F-statistic: 18.45 on 40 and 451 DF,  p-value: < 2.2e-16
```

That suggests spot price of commodity 5 depends on the price of commodity 2 of 8, 5, 2, 1 days ago. Their effects are all positive except the price of commodity 2 of 5 days ago. That means, a higher price of

commodity 1 of 8, 2 ,1 days ago

will denote a higher spot price of commodity 4.

Lower price of commodity 1 of 5 days ago will lead to a lower price of commodity 5.

Appendix