SM_HW2

Kwon Namtaek

2020 11 1

통계학과 2015313693 권남택

```
library(dplyr)
library(ggplot2)
library(gam) # 1 번에서 사용
library(astsa) # 2 번에서 사용
```

필요한 라이브러리 먼저 로드.

1. Consider the data in `house.csv' file. The our goal is to build the best model predicting house price based on some factors in a city.

```
house = data.table::fread('house-1.csv')
```

1-(1). Using the data modelling techniques (parametric modelling), build your model and fit it to the data for attaining the lowest AIC value. To compute AIC in R, use AIC built-in function.

먼저 모든 변수들을 넣고 적합하기 전에, 시간에 관한 date 변수와 공간에 관한 lat, lon 변수를 제외하고 선형회귀를 적합해본다. 기본적으로 고려한 age, dist, store 변수가 괜찮은지, 회귀 가정들은 만족이 되는지를 확인하기 위한 과정이다.

```
lm_fit1 = lm(price ~ age + dist + store, data = house)
summary(lm_fit1)
```

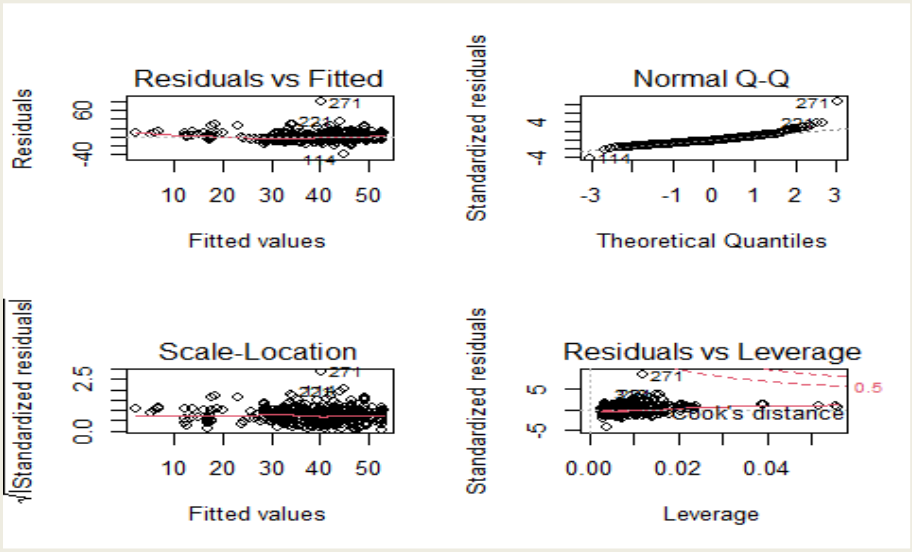
```
##
## Call:
## lm(formula = price ~ age + dist + store, data = house)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.304  -5.430  -1.738   4.325  77.315
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.977286   1.384542  31.041  < 2e-16 ***
## age         -0.252856   0.040105  -6.305 7.47e-10 ***
## dist        -0.005379   0.000453 -11.874  < 2e-16 ***
## store        1.297443   0.194290   6.678 7.91e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.251 on 410 degrees of freedom
## Multiple R-squared:  0.5411, Adjusted R-squared:  0.5377
## F-statistic: 161.1 on 3 and 410 DF,  p-value: < 2.2e-16
```

```
AIC(lm_fit1)
```

```
## [1] 3022.972
```

사용된 변수들은 매우 유의하며, AIC 는 3022.972 이다. 이것이 가장 베이스라인이 되는 모델의 AIC 이므로, 여기서 적절한 전처리들을 동반하며 AIC 를 낮춰가자. 먼저 회귀 가정부터 확인한다.

```
par(mfrow = c(2,2))
plot(lm_fit1)
```




```
par(mfrow = c(1,1))
```

R 에서 제공하는 잔차플랏을 보았을 때 회귀가정이 깨지는 경우들을 확인할 수 있다. 첫번째 Residual vs Fitted 플랏에서는 등분산에는 큰 문제가 없어 보이지만, 잔차에 이차(quadratic)의 형태가 남아있음을 확인할 수 있다. 두번째 Normal QQ plot 을 확인하면, 정규성이 심각하게 깨져서 다소 감마분포와 같이 왼쪽으로 치우친 형태이다. y 변수가 price 가격이므로 이런 형태로 정규성이 깨지는 것은 충분히 예상가능하다. 세번째와 네번째 플랏에서는 특별히 문제점을 파악하기 어렵다. Scale-Location 플랏에서는 등분산이 깨지는 형태를 관측하기 어려우며, Residuals vs Leverage 플랏에서도 이상치(outlier)나 지레점(leverage point)는 확인 가능하나, 영향점으로 분류되지는 않는다.

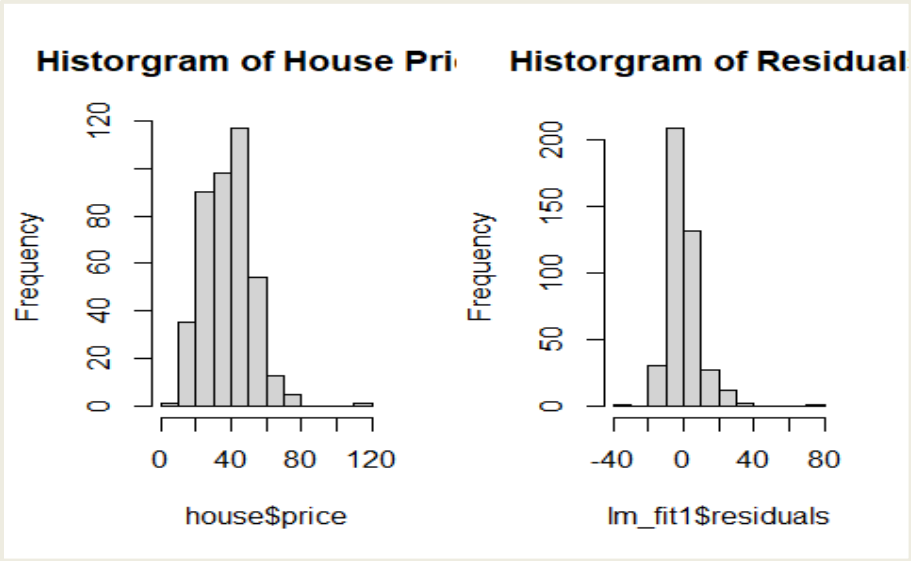
현재 명확하게 정규성 가정은 위배되었지만, 등분산은 깨지지 않은 것같다. 추가적으로 test 를 통해 등분산성, 독립성, 정규성에 대해 확인해보자.

```
car::ncvTest(lm_fit1) # 등분산 기각 불가

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.4014219, Df = 1, p = 0.52636

car::durbinWatsonTest(lm_fit1) # 독립성 기각 불가

##  lag Autocorrelation D-W Statistic p-value
##    1     -0.06059551      2.114929    0.22
##  Alternative hypothesis: rho != 0

shapiro.test(lm_fit1$residuals) # 정규성 기각

##
##  Shapiro-Wilk normality test
##
```

```
## data:  lm_fit1$residuals
## W = 0.87858, p-value < 2.2e-16
```

등분산성과 독립성의 경우에는 검정을 통해 기각할 수 없었다. 하지만 정규성 같은 경우 shapiro-wilk test 를 통해 기각했다. 구체적으로 살펴보자.

```
# 정규성의 확인
par(mfrow = c(1,2))
hist(house$price, main = 'Historgram of House Price')
hist(lm_fit1$residuals, main = 'Historgram of Residuals')
```

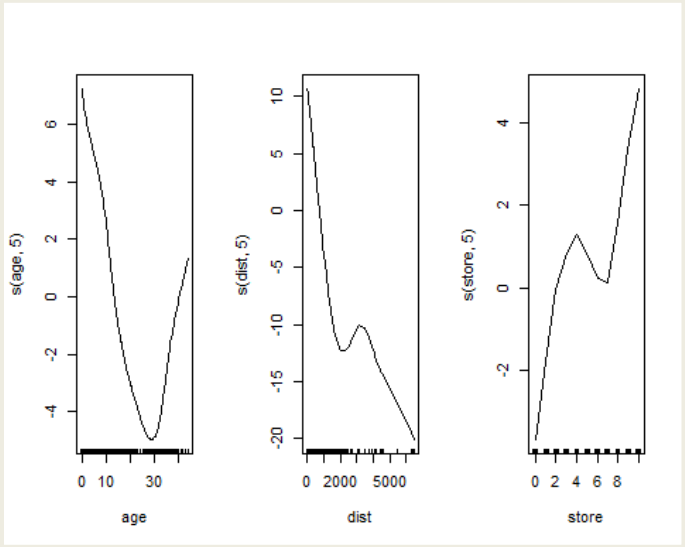



```
par(mfrow = c(1,1))
```

y 와 residual 에서도 확인할 수 있듯, 전체적으로 왼쪽으로 치우친 형태이다. 이를 수정하기 위해 Gamma Regression 형태를 고려할 수 있다. 이를 도입하는 논리를 추후에 설명하려 한다.

여기서 현재 주어진 변수들에 대해 고차항을 추가했을 때 더 좋은 적합이 가능한지 확인하기 위해 Generalized Additive Model 을 고려하려 한다.

```
gam_fit1 = gam(price ~ s(age,5) + s(dist,5) + s(store, 5), data = house)

par(mfrow=c(1,3))
plot(gam_fit1)
```



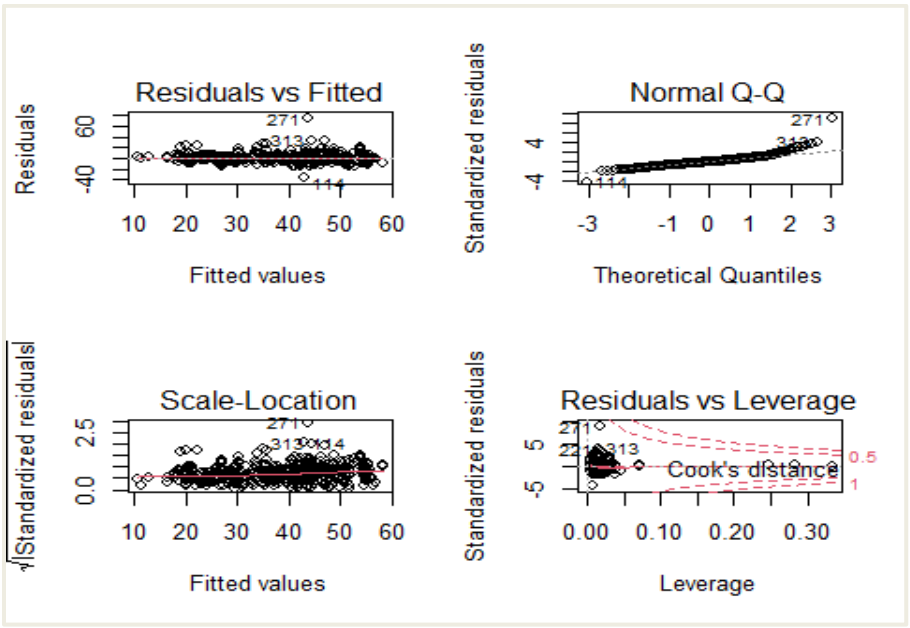

```
par(mfrow = c(1,1))
```

age 변수의 경우 이차(quadratic)의 형태가 비교적 명확하게 드러난다. dist 의 경우 간단히 봐서는 삼차(cubic)의 형태가 나타나지만, 이를 선형으로 충분할지, 혹은 지수적인 감소와 같은 형태로 수정해줘야 할지 추가적인 절차가 필요하다. store 의 경우에도 삼차 형태가 나타나지만 선형으로 충분할지 검증해야 한다.

```
# dist
gam_fit1_3 = gam(price ~ s(age,5) + dist + s(store, 5), data = house)
anova(gam_fit1, gam_fit1_3)

## Analysis of Deviance Table
##
## Model 1: price ~ s(age, 5) + s(dist, 5) + s(store, 5)
## Model 2: price ~ s(age, 5) + dist + s(store, 5)
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1       398      27073
## 2       402      31211 -4  -4137.6 1.944e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# store
gam_fit1_2 = gam(price ~ s(age,5) + s(dist,5) + store, data = house)
anova(gam_fit1, gam_fit1_2)

## Analysis of Deviance Table
##
## Model 1: price ~ s(age, 5) + s(dist, 5) + s(store, 5)
## Model 2: price ~ s(age, 5) + s(dist, 5) + store
##   Resid. Df Resid. Dev      Df Deviance Pr(>Chi)
## 1       398      27073
## 2       402      27566 -4.0003  -493.28   0.1232
```

dist 의 경우 anova 를 통해 귀무가설을 기각했으므로 비선형성을 추가적으로 잡아주는 것이 적절하고, store 의 경우에는 선형으로 유지하는 것이 적절하다. 따라서 추가적으로 비선형성을 고려한 모델을 본다.

```
lm_fit2 = lm(price ~ age + I(age^2) + dist + I(dist^2) + I(dist^3) + store , data = house)
par(mfrow = c(2,2))
plot(lm_fit2)
```

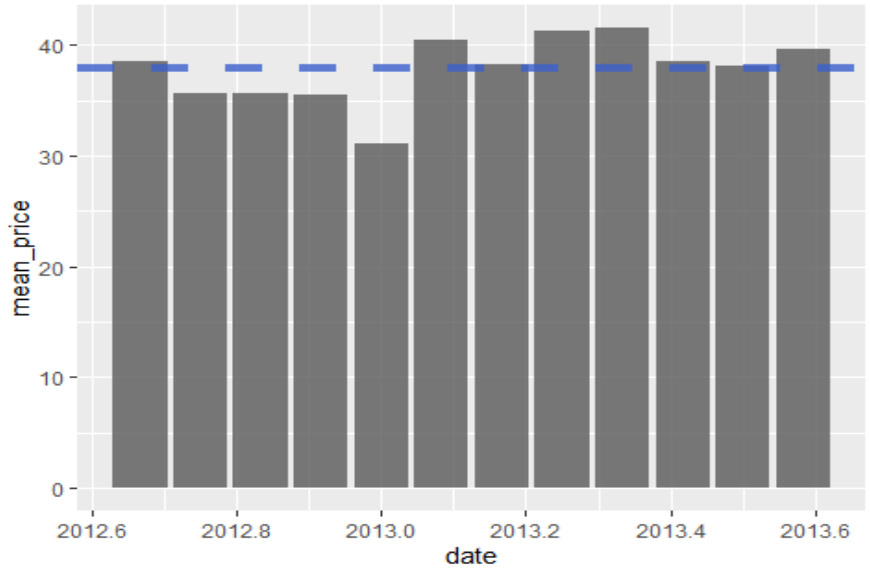



```
par(mfrow = c(1,1))
AIC(lm_fit2) # 2945.848

## [1] 2945.848
```

첫번째 Residual Vs Fitted 플랏을 확인했을 때, 비선형성이 충분하게 사라졌다고 볼 수 있다. AIC 를 통해서도 모델이 나아졌음을 확인 가능하다.

여기서 우리가 아직 사용하지 않은 변수는 date 와 공간에 대한 변수인 lat, lon 이다. date 의 경우에는 1 월부터 12 월 month 에 대한 변수이며, 시간에 따른 자기상관성을 교정하기 위함이라기 보다, month 라는 factor 에 의한 차이가 있는지를 보기 위한 변수로 사용될 수 있다. 이런 아이디어에서 출발해, month 에 따른 효과가 유의미한지 시각화와 ANOVA 를 통해 확인하려 한다.

```
house %>% mutate(data = date %>% factor) %>% group_by(date)
 %>% summarise(mean_price = mean(price)) %>%
   ggplot(aes(x = date, y = mean_price)) + geom_bar(stat = 'identity', alpha = 0.8) +
   geom_hline(yintercept = house$price %>% mean,
              color = 'royalblue3', size = 2, linetype = 'dashed', alpha = 0.8)
```



시각적으로 확인했을 때, month 에 의한 효과는 미미해 보인다. 실제로 ANOVA 를 통해 확인해보면,

```
anova_result = aov(price ~ as.factor(date), data = house)
summary(anova_result)

##                  Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(date)  11   2984   271.3   1.484  0.134
## Residuals       402  73477   182.8
```

실제로 month 에 따른 price 의 차이가 없다라는 귀무가설을 기각할 수 없다. 하지만 다중회귀분석은 다른 변수들이 적합된 상태에서 변수의 효과를 고려하기 때문에, 다른 변수들이 있는 상태에서 month 는 유의할 수 있다.

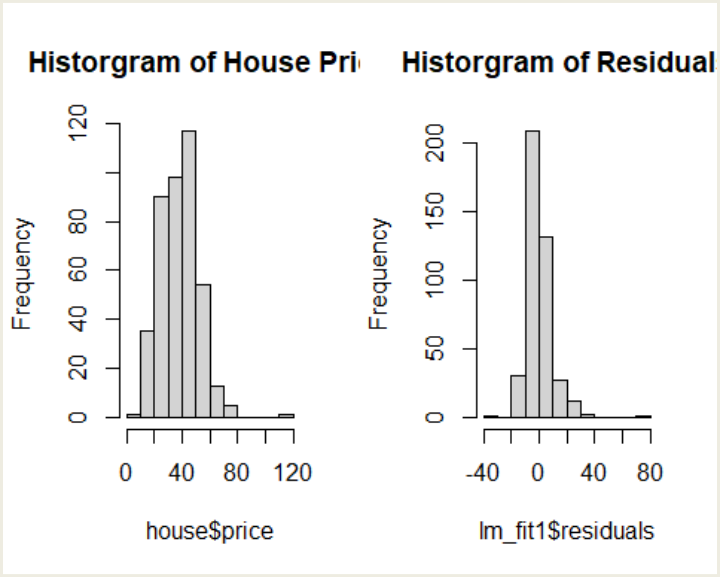
이때, 우리의 목표는 AIC 를 낮추는 것이므로, 12 개의 month 에 대한 가변수를 고려하기 보다, 1 부터 12 의 라벨 인코딩(Label Encoding)으로서 단순 정수 인코딩을 진행한다. 이런 인코딩 방식은 시간이 지남에 따라 가격이 상승하거나 하락하는 패턴이 있다면 더욱 좋은 방법이겠지만, 현재는 그런 추세를 반영하는 것이 아니라, month 라는 변수는 고려하되 변수를 하나만 사용함으로서 단순한 모델을 만들어서 AIC 의 충분한 감소를 만들어내려 한다. month 에 계절성이 있다고 고려할 경우 sin-cos 를 이용한 삼각변환도 고려할 수 있으나, 1)현재 그런 판단이 어려우며 2)실제로 진행했을 때 충분한 AIC 의 감소가 이어지지 않았다.

```
house = house %>% mutate(month = as.numeric(factor(date)))
lm_fit3 = lm(price ~ age + I(age^2) + dist + I(dist^2) + I(dist^3) + store + month,
             data = house)
AIC(lm_fit3) # 2923.27

## [1] 2923.27
```

이제 추가적으로 남은 공간에 관한 변수 lat 과 lon 을 고려할 수 있다. 이를 개별 관측치간의 공관자기상관성을 고려한 모델링을 진행하기 위해서는 SAR (Spatial AutoRegressive Model)을 사용해야 한다. 하지만 현재 우리는 **정규성**이 지켜지지 않은 상황에서 모델링을 진행해왔다. 이런 정규성이 지켜지지 않을 경우, 회귀분석으로 나온 결과들에 대한 해석이 매우 어려워지며, 이후 예측에도 큰 문제가 발생한다. 따라서 공간자기상관성과 정규성사이에서 **정규성**의 해결에 더 큰 목표를 두고 Generalized Linear Model 로 진행하려 한다.

다만, 공간적 특성에 따른 가격이 결정되는 경우도 고려하기 위해, 전체 위경도 상에서 가격들이 어떻게 분포하기 있는지를 확인하여, lon, lat 변수를 Generalized Linear Model 안에서 풀어내려 한다.

```
# 정규성의 확인
par(mfrow = c(1,2))
hist(house$price, main = 'Historgram of House Price')
hist(lm_fit1$residuals, main = 'Historgram of Residuals')
```




```
par(mfrow = c(1,1))
```

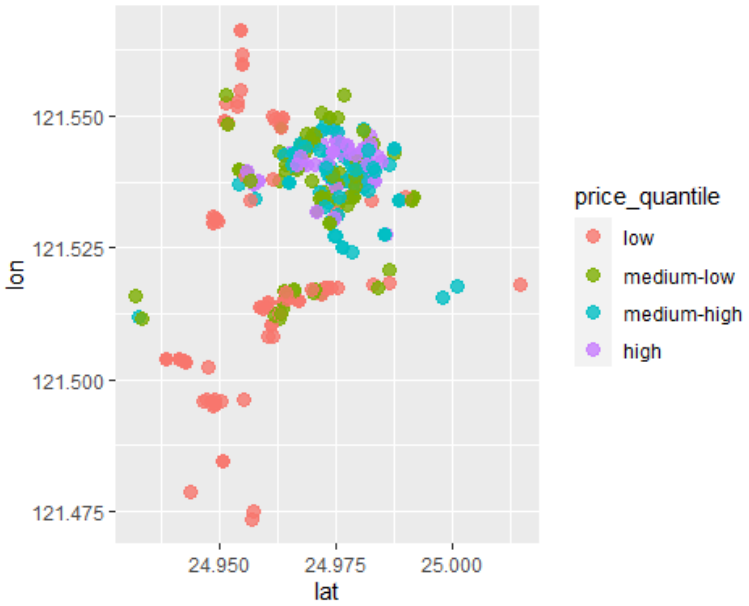
다시한번 히스토그램을 관찰하면, 전체적으로 왼쪽에 치우쳐져 정규성이 깨진 형태이다. 이런 경우, GLM 에서 Gamma Regression 모델을 고려하게 되면, 조금 더 적절한 모델링이 가능하다. 다만 완전히 Gamma Regression 에 가까운 형태는 아니지만, 이를 시도할 수 있는 형태로 존재한다고 판단했고, 추가적으로 이런 모델링을 진행했을 때 더 좋은 AIC 값을 도출할 수 있었다.

```
glm_fit1 = glm(price ~ age + I(age^2) + dist + I(dist^2) + I(dist^3) + store + month,
               data = house, family = Gamma(link = 'inverse'))
AIC(glm_fit1)  # 2881.162
```

```
## [1] 2881.162
```

이제 추가적으로 공간에 따른 효과를 보정해주려 한다. 지도상에서 분표하는 형태에 따라 가격이 어떻게 변하는지 확인하려 한다.

```
quant_cut = quantile(house$price, c(0.25, 0.5, 0.75, 1))
house %>% mutate(price_quantile = cut(price, c(0, quant_cut), c('low', 'medium-low', 'medium-h
igh', 'high'))) %>%
  ggplot(aes(x = lat, y = lon, color = price_quantile)) + geom_point(size = 3, alpha = 0.8)
```



lat 위도의 경우, 가격이 낮은 곳들은 주로 왼쪽에 분포하는 형태가 나타나고, 중심부에서는 medium-low 부터 high 까지 혼재하는 형태이다. 큰 틀에서 볼 경우, 위도가 낮은 공간에서는 저가격을 가지고, 높은 지역에서는 상대적으로 고가격의 집이 위치한다고 매우 거친 요약이 가능하다.

lon 경도의 경우 특징적으로 말하기 어렵다. 일단 가격이 낮은 집들이 경도와 무관하게 존재하고 있다. 중심부에서는 medium-low 부터 high 까지 다양하게 나타나지만, 가장 낮은 경도와 가장 높은 경도에서는 low 위주로 나타난다. 따라서 lon 경도의 경우 충분한 정보를 가지고 있다고 보기 어렵다.

실제로 이 지역에 대해 구글 지도를 통해 확인하였는데, 많은 점들이 모여져 있는 지역의 경우 강 주변에 존재하는 도시 중심부였다. 가격이 낮은 집들이 주로 위치하는 남서부 지역은 도시 외곽지였이었다.

따라서 lat 만 추가적으로 변수로 고려해서 모델링을 진행하면, 또한 추가적으로 store 변수가 다른 변수들이 모두 적합된 상황에서 유의미하지 않다고 결과가 나왔기 때문에 이를 제외해서 적합하면,

```
glm_fit2 = glm(price ~ age + I(age^2) + dist + I(dist^2) + I(dist^3) + month + lat,
               data = house, family = Gamma(link = 'inverse'))
AIC(glm_fit2)  # 2801.67
```

```
## [1] 2801.67
```

최종적인 AIC = 2801.67 이다.

1-(2). Based on the model obtained from part (1), interpret the relationship between house price and each input variable as detail as possible.

```
summary(glm_fit2)
```

```
##
## Call:
## glm(formula = price ~ age + I(age^2) + dist + I(dist^2) + I(dist^3) +
##     month + lat, family = Gamma(link = "inverse"), data = house)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
```

```
## -1.25945  -0.11865  -0.00020   0.09327   1.02276
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.881e+00  7.213e-01   9.540  < 2e-16 ***
## age          5.019e-04  7.909e-05   6.346 5.92e-10 ***
## I(age^2)    -9.281e-06  1.956e-06  -4.744 2.91e-06 ***
## dist         1.624e-05  1.644e-06   9.883  < 2e-16 ***
## I(dist^2)   -4.989e-09  9.355e-10  -5.333 1.61e-07 ***
## I(dist^3)    6.003e-13  1.334e-13   4.499 8.91e-06 ***
## month       -3.833e-04  7.564e-05  -5.067 6.15e-07 ***
## lat         -2.749e-01  2.889e-02  -9.516  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.04099755)
##
##     Null deviance: 58.266  on 413  degrees of freedom
## Residual deviance: 15.991  on 406  degrees of freedom
## AIC: 2801.7
##
## Number of Fisher Scoring iterations: 4
```

현재 사용한 Gamma Regression 의 link 는 inverse 이다. 따라서 기본적으로 나오는 형태를 수식으로 표현하면 다음과 같다.

$$\frac{1}{\mu_i} = {x_i}^T\beta = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$$

이는 mu 의 역수에 대한 해당 결과들의 선형결합이다. 이를 양변에 역수를 취해주게 되면, 다음과 같은 형태가 나타난다.

$$\mu_i = \frac{1}{\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}}$$

y=1/x 의 그래프를 생각하면, x 가 증가함에 따라 y 는 감소함을 알 수 있다. 따라서 summary 에서 양의 부호가 나오는 변수는, 오히려 한 단위 증가함에 따라 평균적인 가격이 떨어진다고 해석할 수 있다. 반면에 음의 부호를 가지는 변수는, 한 단위 증가함에 따라 평균적인 가격이 상승한다고 볼 수 있다. 다만 이에 대해 정확한 단위로서 얼마가 증가한다고 말하는 것은 어렵다.

개별 변수에 대해서 해석해보려한다. age 의 경우 이차형태를 고려해주었는데, age 에 대한 이차식 자체는 위로 볼록한 concave 형태의 그래프이다. 따라서 age 가 올라감에 따라 대응하는 값이 상승하다가 어느순간 다시 감소하는 형태이다. 그러므로 inverse link 를 고려하게 되면, 건물의 연차가 올라갈수록 평균적인 가격이 감소하다가 어느순간 다시 증가한다고 해석할 수 있다.

dist 의 경우 삼차 형태를 고려해주었는데, 전체적으로 우상향 곡선이다. 따라서 inverse link 를 고려하게 되면, 가장 가까운 지하철역으로부터의 거리가 증가할수록, 평균적인 가격은 하락한다고 해석할 수 있다. 중간에 증감이 변화하긴 하지만, 큰 틀에서는 해당 해석이 가능하다.

month 의 경우 부호가 음수이다. 따라서 링크를 고려했을 때, 시간이 지남에 따라서 평균적인 가격이 증가하고 있다고 해석할 수 있다. 각각의 month 를 11 개의 가변수로 고려해주지 않고, 전체적인 추세를 고려해주었기 때문에 과적합을 방지하려 했다.

lat 의 경우 부호가 음수이다. 따라서 링크를 고려했을 때, 위도가 상승함에 따라서 평균적인 가격이 증가하고 있다고 해석할 수 있다.

이 해석의 공통적인 부분은, 증감의 방향과 정도를 알 수는 있지만, 정확한 증감치를 제시할 수는 없다는 점과, 다른 변수들이 적합되어 있을 때의 해석이라는 점이다.

2. Consider the training data in `pm25 tr.csv` and the test data in `pm25 te.csv`. Suppose that our interest is to predict pm 2.5 concentration based on some meteorological factors. In the dataset, the output variable is pm25. The training set has data measured from March, 1st to May, 20th and the test set has data measured from May 21st to May 25th (next 5 days).

```
train = data.table::fread('pm25_tr-1.csv')
```

2-(1). Using the data modelling techniques (parametric modelling), build your best prediction model from the training data. [NOTE: You might need the transformation of variables or variable selection].

```
train %>% head()
```

```
##    month day hour pm25 DEWP TEMP PRES cbwd   Iws
## 1:     3   1    0   69   -4   -3 1026   SE  7.15
## 2:     3   1    1   59   -5   -4 1026   SE 12.07
## 3:     3   1    2   42   -5   -4 1025   SE 15.20
## 4:     3   1    3   35   -5   -4 1025   SE 18.33
## 5:     3   1    4   35   -5   -5 1025   SE 20.12
## 6:     3   1    5   29   -5   -4 1025   SE 23.25
```

```
plot(train$Iws[1:500], type = 'l', ylab = 'Iws', lwd = 2, ylim = c(0,200))
```

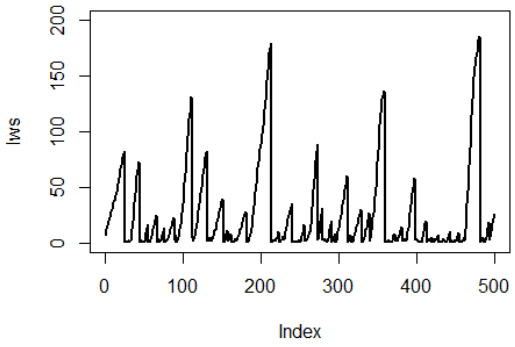
여기서 Iws 변수는 누적된 값이다. 같은 풍향안에서 ws 값이 누적되다가, 풍향이 바뀌면 다시 0 에서 시작하는 형태다. 누적된 형태를 풀어줘서, 해당 시간대가 순수하게 가지는 wind speed 를 구하겠다.

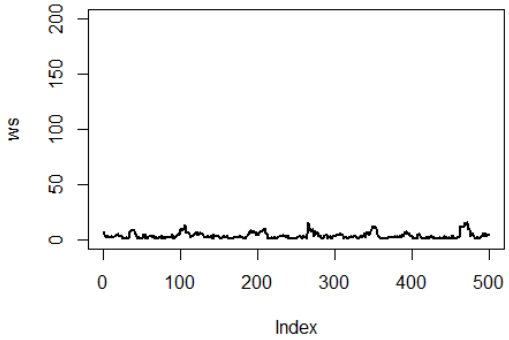
```
train = train %>%
  mutate(Iws_lag = lag(Iws),
         cbwd_lag = lag(cbwd),
         ws = ifelse(cbwd_lag == cbwd, Iws - Iws_lag, Iws))
train[1, 'ws'] = train[1, 'Iws']  # 첫번째 행이 결측치가 되므로 채워준다.
train = train %>% select(-Iws, -Iws_lag, -cbwd_lag)
train %>% head(3) # 확인했을때 문제 없다.
```

```
##    month day hour pm25 DEWP TEMP PRES cbwd   ws
## 1:     3   1    0   69   -4   -3 1026   SE 7.15
## 2:     3   1    1   59   -5   -4 1026   SE 4.92
## 3:     3   1    2   42   -5   -4 1025   SE 3.13
```

바뀐 형태를 확인하면, 잘 변환되었음을 확인할 수 있다.

```
plot(train$ws[1:500], type = 'l', ylab = 'ws', lwd = 2, ylim = c(0, 200))
```





전체적으로 모델링을 진행하고 파생변수, 차수를 결정하기 전에 데이터가 가지는 특징들에 대한 가설을 세우고 이를 확인해가는 과정을 지니려 한다.

1) 먼저 이 데이터는 시간에 따라 관측된 시계열 데이터이다. 기존의 선형회귀분석으로서는 오차의 자기상관이 발생해 모델에 문제가 발생할 수 있다.

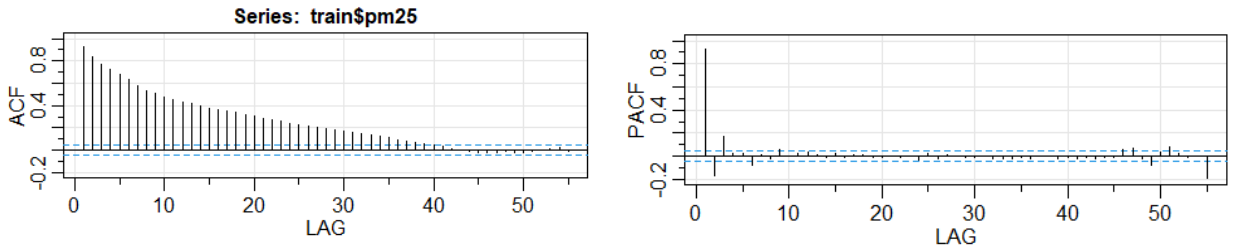
2) 달 내부의 주기성은 없어도, 하루 안에서 주기성이 존재할 수 있다. 만약 이를 고려한다면 더 좋은 모델이 가능해질 수 있다.

3) test 데이터는 5 일만 가지고 있기 때문에, train 에 비해 퍼져있는 정도가 작을 수 있다. 따라서 보다 일반화된 모델을 위해서는 추가적인 validation set 에 대한 분리를 통해 모델을 평가하고 최종적으로 test set 에 대해서 확인해야 한다.

4) PRES(기압)과 ws(풍속) 사이에는 상관관계가 존재함을 상식적으로 알고 있다. 이런 기상데이터들 간에는 상관관계가 존재하므로, 적절한 교호작용항을 추가해주면 모델의 성능이 높아질 것이다.

먼저 1)을 확인해보자. 회귀분석을 진행했을 때 오차항에 대해서는 차후에 확인하고, 자료의 형태를 이해하기 위해 pm25 에 대한 ACF 와 PACF 를 확인한다.

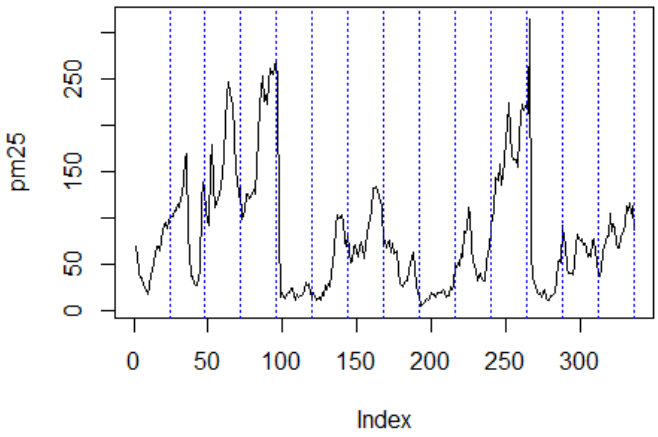
```
acf2(train$pm25)
```



ACF 같은 경우 지수적으로 감소(Exponentially decreasing)하며, PACF 의 경우 3 시점 이후 절단(Cut off)된 형태이다. 따라서 pm25 는 ar(3)의 형태에 가까움을 확인할 수 있는데, 차후에 이를 바탕으로 오차항의 자기상관성을 이해하자.

2)번을 확인하기 위해 pm25 에 대한 플랏을 확인해보자.

```
plot(train$pm25[1:(24*14)], type = 'l', ylab = 'pm25')
for (i in 1:14) {
  abline(v = 24*i, lty = 3, col = 'blue', lwd = 1)
}
```



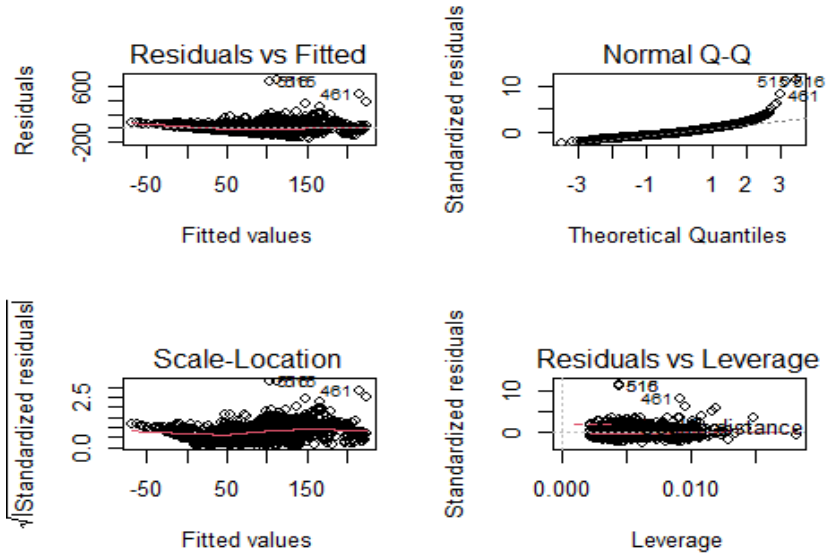
하루 안에서 상승했다가 감소하는 형태가 주기적으로 나타나고 있는 듯도 하지만, 안그런 날도 충분히 많다. 하루 안에서의 경향성이 꾸준히 이어지고 있지는 않다. 따라서 하루 안에서 주기성을 명확하게 관측하기 어려우므로, 이에 대한 sin-cos 을 통한 삼각변환은 고려하지 않아도 되며, 오차항의 계절성은 없다고 조심스럽게 추측할 수 있다.

3)을 고려해서 추가적으로 train set 에 대한 분리를 진행한다. train set 에서 마지막 5 일치를 validation set 으로 진행해서 모델링 과정을 거치자.

```
n = train$pm25 %>% length
val = train[(n-119):n, ]
train_tune = train[1:(n-120), ]
```

제일 기본적인 선형회귀모델부터 적합한다.

```
lm_fit1 = lm(pm25 ~ factor(month) + DEWP + TEMP + PRES + cbwd + ws, data = train_tune)
par(mfrow = c(2,2))
plot(lm_fit1)
```




```
par(mfrow = c(1,1))
```

전체적으로 잔차플랏을 보았을때, 회귀가정들이 깨졌음을 확인가능하다. Residuals vs Fitted 플랏의 경우 점차 분산이 증가하는 형태가 드러나므로 등분산이 깨졌음을 확인 가능하다. Normal QQplot 의 경우 오른쪽으로 꼬리가 긴 형태가 나타나서 정규성이 깨졌다. 아까 pm25 를 확인했을때, 극단적인 값들이 종종 관측되었었기 때문에 이와 관련이 있다. 독립성의 경우 plot 상에서 확인은 어렵지만 깨질것은 예상할 수 있고, 선형성같은 경우 Residuals vs Fitted 을 보았을 때 명확하게 추가적인 적합이 필요한지는 불분명하기 때문에 추가적인 확인이 필요할 것이다. 네번째 Residual vs Leverage 플랏에서는 이상치(outlier)나 지레점(Leverage Point)는 확인가능하나, 영향점은 확인할 수 없다. 추가적으로 테스트를 진행해보겠다.

```
shapiro.test(lm_fit1$residuals) # 정규성 기각

##
##  Shapiro-Wilk normality test
##
## data:  lm_fit1$residuals
## W = 0.85395, p-value < 2.2e-16

car::ncvTest(lm_fit1) # 등분산 기각

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 315.9663, Df = 1, p = < 2.22e-16

car::durbinWatsonTest(lm_fit1, max.lag = 5) # 독립성 기각

##  lag Autocorrelation D-W Statistic p-value
##    1       0.8668333     0.2662087       0
##    2       0.7129388     0.5737631       0
##    3       0.6158383     0.7673445       0
##    4       0.5519238     0.8944132       0
##    5       0.4999817     0.9974583       0
##  Alternative hypothesis: rho[lag] != 0
```

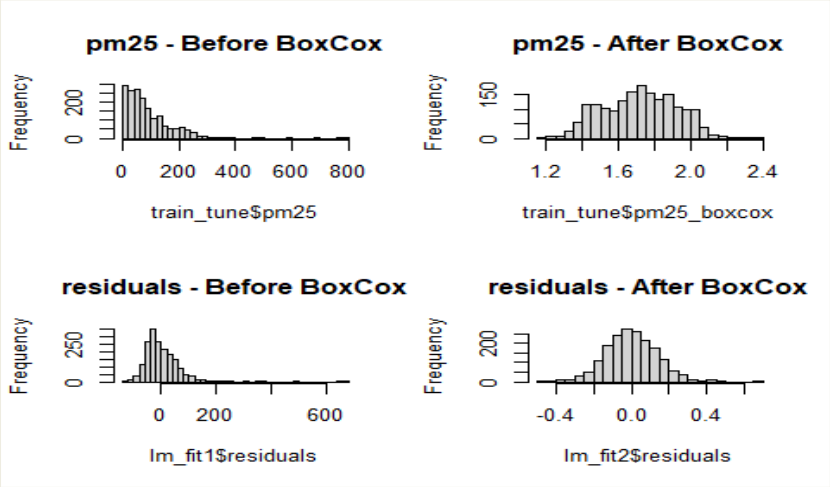
정규성, 등분산성, 독립성 모두 위배되었음을 확인 가능하며, 독립성의 경우 자기상관성이 꾸준하게 감소하는 AR 형태임을 유추할 수 있다.

여기서 먼저 정규성과 등분산성에 대한 수정을 Box-Cox Transformation 으로 해준다.

```
summary(car::powerTransform(train_tune$pm25))

## bcPower Transformation to Normality
##                 Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## train_tune$pm25    0.1269        0.13       0.0789       0.1749
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                             LRT df       pval
## LR test, lambda = (0) 26.89365  1 2.1496e-07
##
## Likelihood ratio test that no transformation is needed
##                             LRT df       pval
## LR test, lambda = (1) 1239.516  1 < 2.22e-16
```

박스콕스를 통해 구한 power 의 값은 0.13 이다. 이를 기준으로 y 를 변환해주고, 변환 전 후를 비교해보겠다.

```r
train_tune = train_tune %>% mutate(pm25_boxcox = pm25^0.13)
lm_fit2 = lm(pm25_boxcox ~ factor(month) + DEWP + TEMP + PRES + cbwd + ws, data = train_tune)

par(mfrow =c(2,2))
hist(train_tune$pm25, main = 'pm25 - Before BoxCox', breaks = 30)
hist(train_tune$pm25_boxcox, main = 'pm25 - After BoxCox', breaks = 30)
hist(lm_fit1$residuals, main = 'residuals - Before BoxCox', breaks = 30)
hist(lm_fit2$residuals, main = 'residuals - After BoxCox', breaks = 30)
```



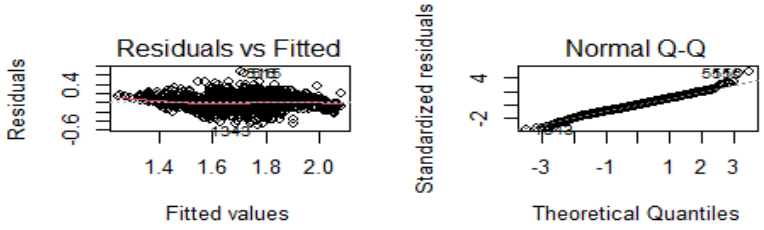

```r
par(mfrow = c(1,1))
```

pm25 를 박스칵스 변환함에 따라, 이전보다 비교적 종 모양(bell shape)에 가까워짐을 확인할 수 있다. 변환된 pm25 의 경우 완전한 종모양이 아니고, 중간에 튀어나온 부분이나 오른쪽의 긴꼬리가 조금은 남아있다. 하지만 적합된 이후의 잔차를 보게 되면, 변환했을때의 잔차는 평균 0 을 중심으로하는 정규분포에 매우 가까움을 확인할 수 있다.

```r
shapiro.test(lm_fit2$residuals) # 정규성 기각

##
##  Shapiro-Wilk normality test
##
## data:  lm_fit2$residuals
## W = 0.99419, p-value = 1.42e-06
```

안타깝게도 변환된 pm25 에 대한 회귀식의 잔차의 정규성은 기각되지만, 잔차의 분포 형태는 정규분포에 가까움을 다시 한번 확인하자. 현재 우리의 표본이 약 1800 개로 작은 수치는 아니다. 따라서 정규분포에서 조금 어긋나는 값이 늘어날수록 귀무가설 정규분포를 따른다를 기각하기 쉬워진다. 따라서 정규성 검정결과에서는 귀무가설을 기각했지만, 데이터의 분포를 확인했을 때 정규분포에 이전보다 훨씬 근접하므로 추가적인 처치를 시행하지는 않는다. 총체적으로 잔차플랏에서 다시 확인해보겠다.

```r
par(mfrow = c(2,2))
plot(lm_fit2)
```



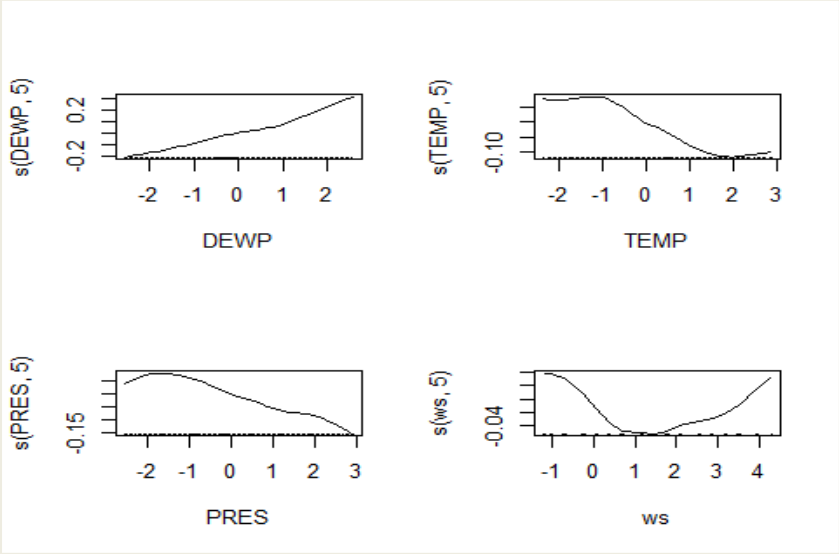

```
par(mfrow = c(1,1))
```

첫번째 Residuals vs Fitted 플랏에서 이분산이 상당히 완화되었으며, Normal QQplot 에서도 오른쪽으로 꼬리가 길긴 하지만 이전보다 정규성에 근접함을 알 수있다. 등분산성과 독립성에 대한 검정을 추가적으로 시행하면,

```
car::ncvTest(lm_fit2)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1.248221, Df = 1, p = 0.26389

car::durbinWatsonTest(lm_fit2, max.lag = 5)

##  lag Autocorrelation D-W Statistic p-value
##    1       0.8843236     0.2312997       0
##    2       0.7792779     0.4412052       0
##    3       0.6877271     0.6234328       0
##    4       0.6041315     0.7893320       0
##    5       0.5272911     0.9415122       0
##  Alternative hypothesis: rho[lag] != 0
```

검정을 통해 등분산을 만족하고 있음을 확인 가능하고, 독립성의 경우 여전히 만족하지 못하고 있다. **독립성의 경우 이후에 추가적으로 수정해주겠다.** 먼저 고차항을 고려하겠다. 고차항을 고려해줄 경우, 현재 PRES 와 같이 단위때문에 제곱에 민감한 변수들이 존재하므로, scaling 을 진행해준다.

```
DEWP_mean = mean(train_tune$DEWP); DEWP_sd = sd(train_tune$DEWP)
TEMP_mean = mean(train_tune$TEMP); TEMP_sd = sd(train_tune$TEMP)
PRES_mean = mean(train_tune$PRES); PRES_sd = sd(train_tune$PRES)
ws_mean = mean(train_tune$ws); ws_sd = sd(train_tune$ws)

train_tune = train_tune %>% mutate(DEWP = scale(DEWP) %>% as.vector,
                                   TEMP = scale(TEMP) %>% as.vector,
                                   PRES = scale(PRES) %>% as.vector,
                                   ws = scale(ws) %>% as.vector)

gam_fit1 = gam(pm25_boxcox ~ s(DEWP,5) + s(TEMP,5) + s(PRES,5) + s(ws,5) + cbwd, data = train_
tune)
par(mfrow = c(2,2))
plot(gam_fit1)
```




```
par(mfrow = c(1,1))
```

DEWP 를 제외하고는 모두 고차항을 고려해봄직 하다. ws 는 비교적 명확하므로, 다른 TEMP 와 PRES 를 고려해준다.

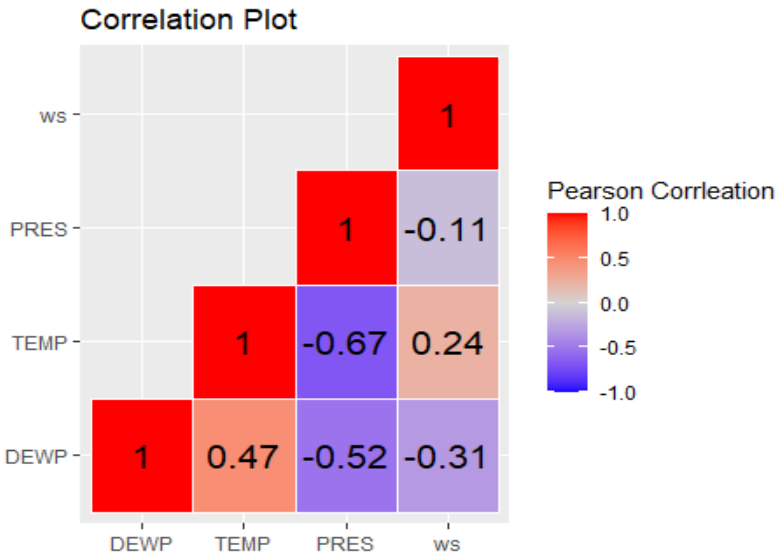
```
gam_fit2 = gam(pm25_boxcox ~ DEWP + s(TEMP,5) + s(PRES,5) + cbwd + s(ws,5), data = train_tune)

gam_fit3 = gam(pm25_boxcox ~ DEWP + s(TEMP,5) + PRES + cbwd + s(ws,5), data = train_tune)
anova(gam_fit2, gam_fit3)

## Analysis of Deviance Table
##
## Model 1: pm25_boxcox ~ DEWP + s(TEMP, 5) + s(PRES, 5) + cbwd + s(ws, 5)
## Model 2: pm25_boxcox ~ DEWP + s(TEMP, 5) + PRES + cbwd + s(ws, 5)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      1804     40.403
## 2      1808     40.716 -4 -0.31218 0.007493 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

gam_fit4 = gam(pm25_boxcox ~ DEWP + TEMP + s(PRES,5) + cbwd + s(ws,5), data = train_tune)
anova(gam_fit2, gam_fit4)

## Analysis of Deviance Table
##
## Model 1: pm25_boxcox ~ DEWP + s(TEMP, 5) + s(PRES, 5) + cbwd + s(ws, 5)
## Model 2: pm25_boxcox ~ DEWP + TEMP + s(PRES, 5) + cbwd + s(ws, 5)
##   Resid. Df Resid. Dev      Df Deviance  Pr(>Chi)
## 1      1804     40.403
## 2      1808     41.032 -4.0001 -0.62898 1.199e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

귀무가설이 기각되었으므로, TEMP 와 PRES 에 대한 비선형 관계를 잡아주는 것이 합리적임을 확인할 수 있다. ws, TEMP, PRES 에서 이차항을 고려해준다. TEMP 에서 삼차항을 고려할수도 있었지만, 실제 성능측면에서 이차항으로 충분했다.

lm_fit3 = lm(pm25_boxcox ~ factor(month) + DEWP + TEMP + I(TEMP^2) + PRES + I(PRES^2) + cbwd + ws + I(ws^2), data = train_tune)

여기서 추가적으로 교호작용항도 고려해주자.

```
cor_mat = train_tune %>% select(DEWP, TEMP, PRES, ws) %>% cor
cor_mat[upper.tri(cor_mat)] = NA
melted_cormat = data.table::melt(cor_mat, na.rm=T)

melted_cormat %>% ggplot(aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(value,2)), size = 5.5) +
  scale_fill_gradient2('Pearson Corrleation', mid = "lightgrey", low = "blue", high = "red",
                       midpoint = 0, limit = c(-1,1))  +
  labs(x = "", y = "", title = "Correlation Plot")
```



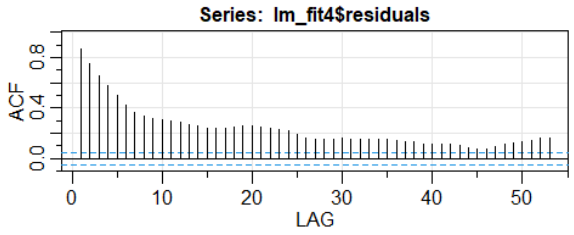
상관계수 플랏을 참고하여, 실제 교호작용항에 대한 변수 선택을 했을 때, 다음의 결과가 가장 성능이 좋았다.

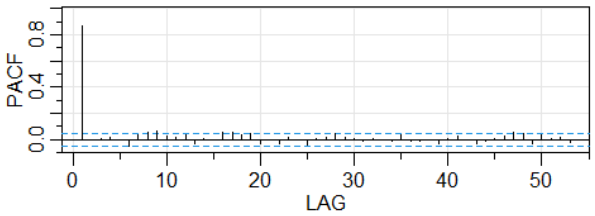
```
lm_fit4 = lm(pm25_boxcox ~ factor(month) + DEWP + TEMP + I(TEMP^2) + PRES + I(PRES^2) + cbwd +
 ws + I(ws^2) +
               DEWP:TEMP + TEMP:PRES, data = train_tune)

pred_lm_fit4 = predict(lm_fit4, val %>% mutate(DEWP = (DEWP - DEWP_mean)/DEWP_sd,
                                               TEMP = (TEMP - TEMP_mean)/TEMP_sd,
                                               PRES = (PRES - PRES_mean)/PRES_sd,
                                               ws = (ws - ws_mean)/ws_sd))
sum(((val$pm25 - pred_lm_fit4^(1/0.13))^2)/length(pred_lm_fit4))  # 2971.757

## [1] 2971.757
```

현재 정규성과 등분산성, 선형성에 대한 수정을 마치고, 교호작용항까지 고려해주었다. 이때 validation set 에 대한 mse 는 2971 이다.

하지만 여기서 가장 중요한 **독립성**은 아직 해결하지 못했다. 독립성을 모델로서 해결하는 과정을 다음에 담으려 한다.

```
acf2(lm_fit4$residuals)
```





ACF 는 지수적으로 감소하며, PACF 는 1 시점 이후로 절단된 형태이기 때문에, 오차항은 AR(1)을 따른다고 볼 수 있다. 따라서 적절한 모델링을 위해서는, 오차항의 자기상관성을 AR(1)으로 잡고, 해당 공분산 행렬의 구조를 최소제곱과정에서 고려해주는 일반화 최소제곱법 (GLS : Generalized Least Square)을 고려해야 한다.

```
train_dummy = train_tune %>% mutate(cbwdNE = ifelse(cbwd == 'NE', 1, 0),
                                    cbwdNW = ifelse(cbwd == 'NW', 1, 0),
                                    cbwdSE = ifelse(cbwd == 'SE', 1, 0),
                                    month3 = ifelse(month == 3, 1, 0),
                                    month4 = ifelse(month == 4, 1, 0),
                                    TEMP_quad = TEMP^2, PRES_quad = PRES^2,
                                    ws_quad = ws^2,
                                    DEWPxTEMP = DEWP*TEMP, TEMPxPRES = TEMP*PRES)
#직접 최적화 윈해 가변수 넣어줌
X = cbind(1,train_dummy %>% select(DEWP, TEMP, TEMP_quad, PRES, PRES_quad,
                                   ws, ws_quad, month3, month4,
                                   cbwdNE, cbwdNW, cbwdSE, DEWPxTEMP, TEMPxPRES)) %>% as.matri
x()
Y = train_tune$pm25_boxcox
n = length(Y)
S = diag(rep(1,n))    # initial covariance matrix
p = dim(X)[2]

mdif = 100000
beta.old = rep(100000, 15)
I = 0
```

가변수를 잡아주고, 고차항과 교호작용항을 직접 넣어주어서, AR(1)구조를 잡아내는 회귀모형을 만들어낼것이다.

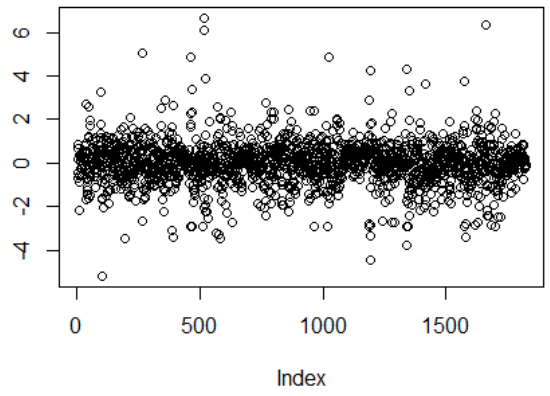
```
while(mdif > 0.00000001) {
  hat_mat = (t(X) %*% solve(S) %*% X) %>% eigen
  beta.new = as.vector((hat_mat$vectors %*% diag(hat_mat$values^(-1), p) %*% t(hat_mat$vector
s)) %*%t(X) %*% solve(S) %*% Y)
  # 역행렬을 안정적으로 구할 수 있도록 일반화 역행렬을 넣어주었다.
  r = as.vector(Y - (X %*% beta.new))
  ar1 = sarima (r, 1,0,0, no.constant=T, details=F)
  alpha = ar1$fit$coef
  sigma2 = ar1$fit$sigma2

  mdif = max(abs(beta.new - beta.old))
  beta.old = beta.new
  I = I + 1
  print(paste('I is ', I, ' & mdif is ', mdif))

  # Construct covariance matrix
  S = matrix(nrow=n,ncol=n)
  for (i in 1:n)
  {
    for (j in 1:n)
```

```
    {
      if (i == j) S[i,j] = 1
      if (i != j) S[i,j] = alpha^(abs(i-j))
    }
  }
  S = (sigma2 / (1-alpha^2)) * S
}
```

```
## [1] "I is  1  & mdif is  100000.066601325"
## [1] "I is  2  & mdif is  0.12074326418621"
## [1] "I is  3  & mdif is  0.0666052322374459"
## [1] "I is  4  & mdif is  0.00930840373518463"
## [1] "I is  5  & mdif is  0.00101140349321674"
## [1] "I is  6  & mdif is  0.000109245750013226"
## [1] "I is  7  & mdif is  1.17971596134692e-05"
## [1] "I is  8  & mdif is  1.27395181381829e-06"
## [1] "I is  9  & mdif is  1.37495874852345e-07"
## [1] "I is  10  & mdif is  1.49232928237097e-08"
## [1] "I is  11  & mdif is  1.91236540492135e-09"
```

계산과정에서 t(X) %*% solve(S) %*% X 가 input 에 따라 역행렬이 존재하지 않는 경우가 있었기 때문에, 일반화 역행렬을 통해 계산했다. 전체적인 과정은 example 의 과정과 유사하다.

```
S_eigen = S %>% eigen
residual_vector = (t(Y - (X %*% beta.new)) %*% (S_eigen$vectors %*% diag(S_eigen$values^(-1/
2)) %*% t(S_eigen$vectors))) %>% as.vector()
residual_vector %>% plot
```



잔차에 명확한 패턴이 보이지 않지만, 직접 test 를 통해 확인해야할 필요성은 존재한다. durbin watson statistic 을 유도해서 확인한다.

```
residual_diff = residual_vector[2:length(residual_vector)] - residual_vector[1:(length(residua
l_vector)-1)]
dw_statistics = sum(residual_diff^2) / sum(residual_vector^2); dw_statistics
```

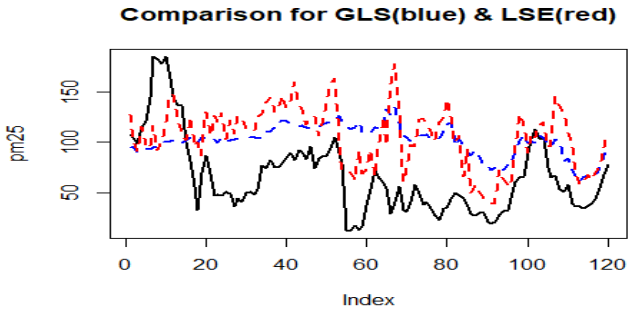
```
## [1] 1.764972
```

더빈왓슨 통계량은 1.76 로, 2 에 근접하기 때문에 기존의 자기 상관성이 상당부분 교정되었음을 확인가능하다. 물론 여전히 일부 자기상관이 남아 있는 형태이긴 하다. 이런 GLS 의 결과를 LSE 와 비교해보자.

```r
# Prediction by GLS
val_dummy = val %>% mutate(DEWP = (DEWP - DEWP_mean)/DEWP_sd,
                           TEMP = (TEMP - TEMP_mean)/TEMP_sd,
                           PRES = (PRES - PRES_mean)/PRES_sd,
                           ws = (ws - ws_mean)/ws_sd,
                           month3 = ifelse(month == 3, 1, 0),
                           month4 = ifelse(month == 4, 1, 0),
                           cbwdNE = ifelse(cbwd == 'NE', 1, 0),
                           cbwdNW = ifelse(cbwd == 'NW', 1, 0),
                           cbwdSE = ifelse(cbwd == 'SE', 1, 0),
                           TEMP_quad = TEMP^2, PRES_quad = PRES^2, ws_quad = ws^2,
                           DEWPxTEMP = DEWP*TEMP, TEMPxPRES = TEMP*PRES)
X_val = cbind(1,val_dummy %>% select(DEWP, TEMP, TEMP_quad, PRES, PRES_quad,
                                     ws, ws_quad, month3, month4,
                                     cbwdNE, cbwdNW, cbwdSE, DEWPxTEMP, TEMPxPRES)) %>% as.mat
rix()
y_val = val$pm25
sum((y_val - (X_val %*% beta.new)^(1/0.13))^2)/length(y_val) # 2749

## [1] 2749.33

# Prediction by LSE
pred_lm_fit4 = predict(lm_fit4, val %>% mutate(DEWP = (DEWP - DEWP_mean)/DEWP_sd,
                                               TEMP = (TEMP - TEMP_mean)/TEMP_sd,
                                               PRES = (PRES - PRES_mean)/PRES_sd,
                                               ws = (ws - ws_mean)/ws_sd))
sum(((val$pm25 - pred_lm_fit4^(1/0.13))^2)/length(pred_lm_fit4))  # 2971.757

## [1] 2971.757
```

GLS 와 LSE 방법을 비교했을 때, GLS 의 MSE 가 조금 더 낮음을 확인할 수 있다. GLS 의 경우 공분산 구조를 고려하기 때문에 다시 **BLUE** 가 되기 때문에 해석측면에서 매우 타당한 방법인데, 이경우 GLS 의 성능이 조금 더 좋으므로, 매우 이상적인 상황이다.

```r
plot(val$pm25, type = 'l', lwd = 2,
main = 'Comparison for GLS(blue) & LSE(red)', ylab = 'pm25')
lines((X_val %*% beta.new)^(1/0.13), lwd = 2, lty = 2, col = 'blue')  # GLS fit
lines(pred_lm_fit4^(1/0.13), lwd = 2, lty = 2, col = 'red')  # LSE fit
```



GLS 의 경우 매우 안정적인 그래프이다. 아마 평균주위에서 있다가, 일정 시점이후부터 전체적인 추세에 반응하는 것 같다. 반면 LSE 는 처음부터 변동이 어느정도 존재하고, 실제로 pm25 의 움직임을 따라가려고 움직이나 전체적으로 **과대예측**하는 경향이 있다. 그러나 현재 validation 과 test 가 유사한 움직임이지 않을테니, 우리의 최종모델이 GLS 이더라도 실제 test 에 대해서는 좋지 않은 성능을 발휘할 수도 있다.

전체 train set 에 대해 다시 회귀모형을 만든다.

```
DEWP_mean = mean(train$DEWP); DEWP_sd = sd(train$DEWP)
TEMP_mean = mean(train$TEMP); TEMP_sd = sd(train$TEMP)
PRES_mean = mean(train$PRES); PRES_sd = sd(train$PRES)
ws_mean = mean(train$ws); ws_sd = sd(train$ws)

train = train %>% mutate(pm25_boxcox = pm25^0.13,
                         DEWP = scale(DEWP) %>% as.vector,
                         TEMP = scale(TEMP) %>% as.vector,
                         PRES = scale(PRES) %>% as.vector,
                         ws = scale(ws) %>% as.vector)

train_dummy = train %>% mutate(cbwdNE = ifelse(cbwd == 'NE', 1, 0),
                               cbwdNW = ifelse(cbwd == 'NW', 1, 0),
                               cbwdSE = ifelse(cbwd == 'SE', 1, 0),
                               month3 = ifelse(month == 3, 1, 0),
                               month4 = ifelse(month == 4, 1, 0),
                               TEMP_quad = TEMP^2, PRES_quad = PRES^2,
                               ws_quad = ws^2,
                               DEWPxTEMP = DEWP*TEMP, TEMPxPRES = TEMP*PRES)
#직접 최적화 윈해 가변수 넣어줌
X = cbind(1,train_dummy %>% select(DEWP, TEMP, TEMP_quad, PRES, PRES_quad,
                                   ws, ws_quad, month3, month4,
                                   cbwdNE, cbwdNW, cbwdSE, DEWPxTEMP, TEMPxPRES)) %>% as.matri
x()

Y = train$pm25_boxcox
n = length(Y)
S = diag(rep(1,n))    # initial covariance matrix
p = dim(X)[2]

mdif = 100000
beta.old = rep(100000, 15)
I = 0

while(mdif > 0.00000001) {
  hat_mat = (t(X) %*% solve(S) %*% X) %>% eigen
  beta.new = as.vector((hat_mat$vectors %*% diag(hat_mat$values^(-1), p) %*% t(hat_mat$vector
s)) %*%t(X) %*% solve(S) %*% Y)
  # 역행렬을 안정적으로 구할 수 있도록 일반화 역행렬을 넣어주었다.
  r = as.vector(Y - (X %*% beta.new))
  ar1 = sarima (r, 1,0,0, no.constant=T, details=F)
  alpha = ar1$fit$coef
  sigma2 = ar1$fit$sigma2

  mdif = max(abs(beta.new - beta.old))
  beta.old = beta.new
  I = I + 1
  print(paste('I is ', I, ' & mdif is ', mdif))

  # Construct covariance matrix
  S = matrix(nrow=n,ncol=n)
  for (i in 1:n)
  {
    for (j in 1:n)
    {
      if (i == j) S[i,j] = 1
```

```
      if (i != j) S[i,j] = alpha^(abs(i-j))
    }
  }
  S = (sigma2 / (1-alpha^2)) * S
}
```

```
## [1] "I is  1  & mdif is  100000.067177399"
## [1] "I is  2  & mdif is  0.119829687843485"
## [1] "I is  3  & mdif is  0.0627003775068473"
## [1] "I is  4  & mdif is  0.00871648754632912"
## [1] "I is  5  & mdif is  0.000989351823107526"
## [1] "I is  6  & mdif is  0.000112419185410662"
## [1] "I is  7  & mdif is  1.27806721791912e-05"
## [1] "I is  8  & mdif is  1.45337819035252e-06"
## [1] "I is  9  & mdif is  1.65247091898912e-07"
## [1] "I is  10  & mdif is  1.84874874298302e-08"
## [1] "I is  11  & mdif is  2.09134609630013e-09"
```

2-(2). Compute the test MSE of your model obtained in part (1) using the test set.

test 에 대해서 train 에 적용했던 것과 동일한 전처리를 거치고 test MSE 를 도출하겠다.

```
test = data.table::fread('pm25_te-1.csv')

# ws 만들기
test = test %>%
  mutate(Iws_lag = lag(Iws),
         cbwd_lag = lag(cbwd),
         ws = ifelse(cbwd_lag == cbwd, Iws - Iws_lag, Iws))
test[1, 'ws'] = 3.13  # train 의 마지막 관측치 넣어줌
test = test %>% select(-Iws, -Iws_lag, -cbwd_lag)
test %>% head(4)
```

```
##    month day hour pm25 DEWP TEMP PRES cbwd   ws
## 1:     5  21    0   93    9   24 1006   SE 3.13
## 2:     5  21    1   91   10   22 1007   SE 1.79
## 3:     5  21    2   96   10   22 1007   SE 1.79
## 4:     5  21    3  106   10   20 1007   SE 1.79
```

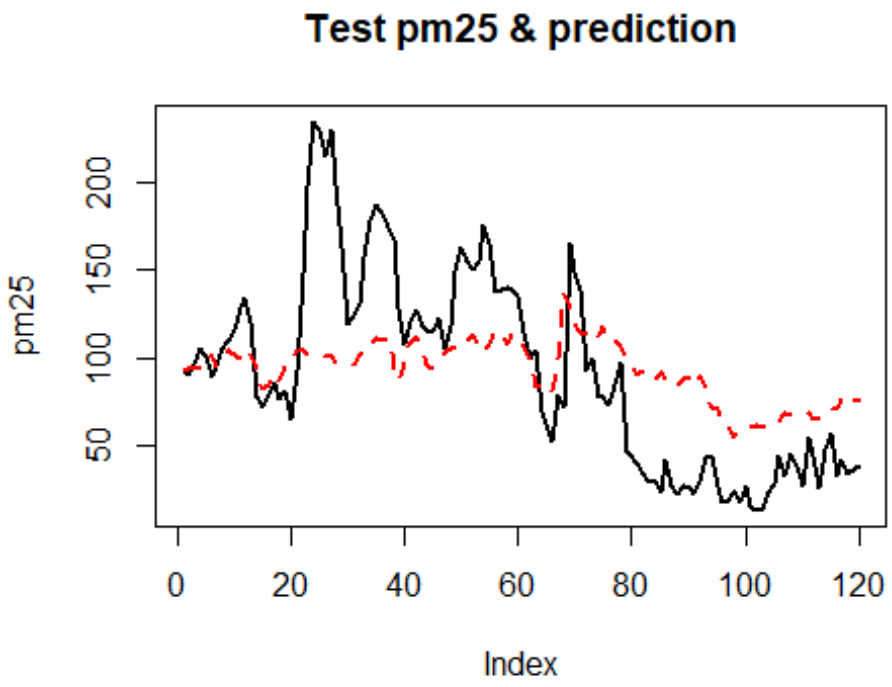
누적 wind speed 를 풀어주고, train 과 동일한 기준으로 scaling 해준다. 그리고 예측결과를 확인한다.

```
test_dummy = test %>%mutate(DEWP = (test$DEWP - DEWP_mean)/DEWP_sd,
                            TEMP = (test$TEMP - TEMP_mean)/TEMP_sd,
                             PRES = (test$PRES - PRES_mean)/PRES_sd,
                             ws = (test$ws - ws_mean)/ws_sd,
                             month3 = ifelse(month == 3, 1, 0),
                             month4 = ifelse(month == 4, 1, 0),
                             cbwdNE = ifelse(cbwd == 'NE', 1, 0),
                             cbwdNW = ifelse(cbwd == 'NW', 1, 0),
                             cbwdSE = ifelse(cbwd == 'SE', 1, 0),
                             TEMP_quad = TEMP^2, PRES_quad = PRES^2, ws_quad = ws^2,
                             DEWPxTEMP = DEWP*TEMP, TEMPxPRES = TEMP*PRES)
X_test = cbind(1,test_dummy %>% select(DEWP, TEMP, TEMP_quad, PRES, PRES_quad,
                                       ws, ws_quad, month3, month4,
                                      cbwdNE, cbwdNW, cbwdSE, DEWPxTEMP, TEMPxPRES)) %>%
         as.matrix()
y_test = test$pm25
sum((y_test - (X_test %*% beta.new)^(1/0.13))^2)/length(y_test) # 2056.816
```

```
## [1] 2056.816
```

Test MSE 는 2056.816 로, 이전의 Validation MSE 로 나온 결과보다 훨씬 좋다. 이전 validation set 보다 비교적 안정적인 형태이기 때문이라고 추측할 수 있다.

```
plot(test$pm25, type = 'l', lwd = 2, main = 'Test pm25 & prediction', ylab = 'pm25')
lines((X_test %*% beta.new)^(1/0.13), lwd = 2, lty = 2, col = 'red')  # LSE fit
```



Test MSE 는 2056.816