Assignment #3

Climate Change

20190552 손지영

사용 data

- ClimateChange.csv : 1983년 5월부터 2008년 12월까지의 지구의 평균적인 대기 질 및 기후와 관련된 월간 데이터

1. Climate Change

[데이터 전처리]

```
# 사용할 패키지 추가
library(psych)
library(ggcorrplot)
library(colorspace)
library(ggplot2)
library(tidyr)
library(leaps)
library(caret)
library(glmnet)
library(rsample)
```

```
# 파일 불러오기
clim = read.csv('ClimateChange.csv')
str(clim)
```

```
## 'data.frame':    308 obs. of  11 variables:
##  $ Year    : int  1983 1983 1983 1983 1983 1983 1983 1983 1984 1984 ...
##  $ Month   : int  5 6 7 8 9 10 11 12 1 2 ...
##  $ MEI     : num  2.556 2.167 1.741 1.13 0.428 ...
##  $ CO2     : num  346 346 344 342 340 ...
##  $ CH4     : num  1639 1634 1633 1631 1648 ...
##  $ N2O     : num  304 304 304 304 304 ...
##  $ CFC.11  : num  191 192 193 194 194 ...
##  $ CFC.12  : num  350 352 354 356 357 ...
##  $ TSI     : num  1366 1366 1366 1366 1366 ...
##  $ Aerosols: num  0.0863 0.0794 0.0731 0.0673 0.0619 0.0569 0.0524 0.0486 0.0451 0.0416 ...
##  $ Temp    : num  0.109 0.118 0.137 0.176 0.149 0.093 0.232 0.078 0.089 0.013 ...
```

```
# year와 month 열 제거
x = clim[, -c(1,2)]
str(x)
```

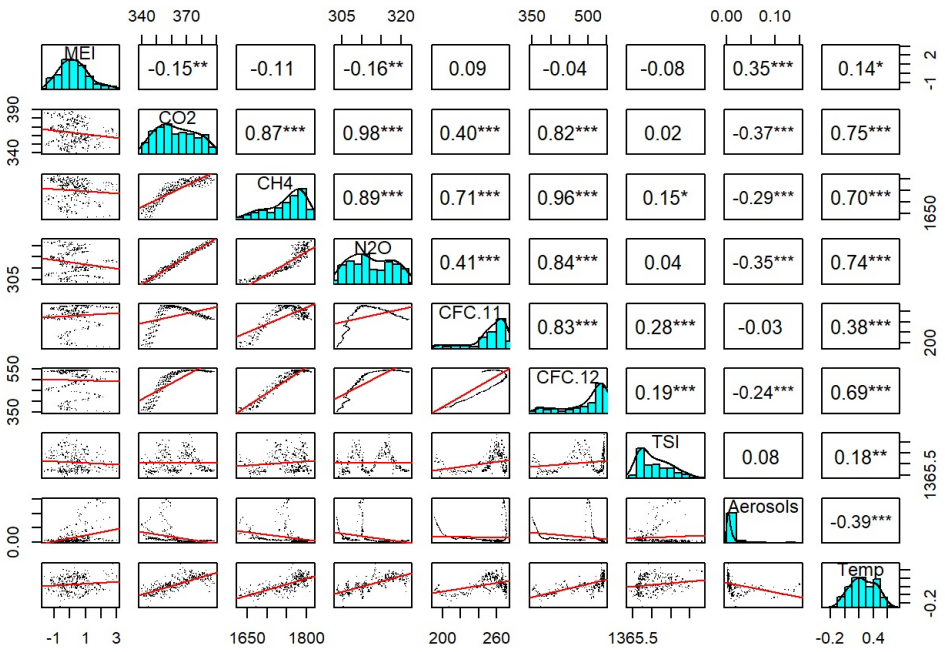
```
## 'data.frame':    308 obs. of  9 variables:
##  $ MEI     : num  2.556 2.167 1.741 1.13 0.428 ...
##  $ CO2     : num  346 346 344 342 340 ...
##  $ CH4     : num  1639 1634 1633 1631 1648 ...
##  $ N2O     : num  304 304 304 304 304 ...
##  $ CFC.11  : num  191 192 193 194 194 ...
##  $ CFC.12  : num  350 352 354 356 357 ...
##  $ TSI     : num  1366 1366 1366 1366 1366 ...
##  $ Aerosols: num  0.0863 0.0794 0.0731 0.0673 0.0619 0.0569 0.0524 0.0486 0.0451 0.0416 ...
##  $ Temp    : num  0.109 0.118 0.137 0.176 0.149 0.093 0.232 0.078 0.089 0.013 ...
```

1.1번

Year 및 Month를 제외한 9개의 변수들 간의 상관 관계를 다양한 그래프를활용하여 시각화해보고, 이로부터 데이터의 특성을 분석해보자.

[변수들간의 분포]

```
# 모든 변수의 분포와 상관관계 살펴보기
pairs.panels(x, lm=TRUE, ellipses=FALSE, rug=FALSE,
             stars=TRUE, pch='.')
```



먼저 target 변수인 Temp와 다른 변수들을 비교해보도록 한다. 위의 출력된 그림을 통해 변수 Aersolos는 Temp 변수와 음의 상관관계를, 나머지 변수들은 모두 양의 상관관계를 갖음을 바로 확인할 수 있다. 추가적으로 양의 상관관계를 갖는 변수들 중 CO2와 N2O와 같은 경우 추세선을 중심으로 점들이 더 모여있다. 또한 숫자로 표시된 상관계수를 살펴보더라도 상대적으로 높은 상관관계를 갖음을 확인할 수 있다. 하지만 변수 MEI와 TSI는 타겟변수인 Temp와 양의 상관관계를 갖지만 상대적으로 약한 것을 점들의 분포와 상관계수를 통해 알 수 있다. 추가적으로 상관계수 옆 별을 통해 해당 관계가 유의한지 아닌지를 알 수 있다. 해당 상관관계가 다른 변수보다 약하지만 그럼에도 유의함을 알 수 있다.

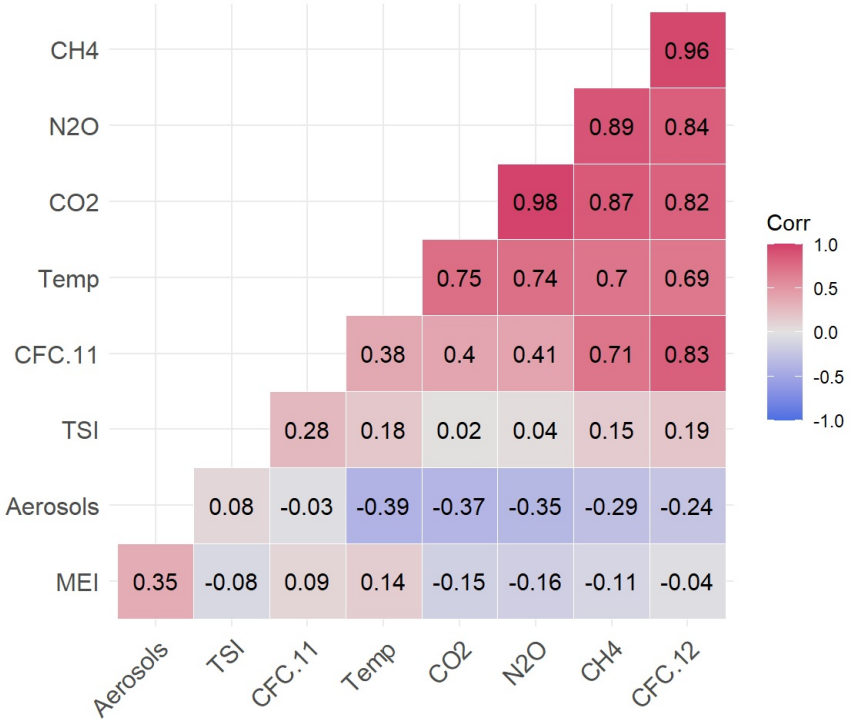
다음으로 변수들끼리의 상관관계를 살펴보도록 하겠다. 먼저 추세선과 점들의 분포를 통해 강한 상관관계를 갖는 변수들이 존재함을 알 수 있다. 예시로 CO2와 N2O의 경우 상관관계가 0.98로 아주 높다. 또한 CH4와 CFC.12, CH4와 N2O, CFC.11와 CFC.12 등과 같은 조합에서 높은 상관관계를 확인할 수 있다. 특히 변수 CFC.12와 같은 경우 다른 4개의 변수와의 상관관계가 0.8이 넘는다. 이를 통해 해당 데이터의 변수들간의 상관관계가 높은 것들이 있으며, 이는 모델의 예측 성능에 좋지 않으므로 적절한 조치가 필요할 것으로 보인다.

[히트맵]

```
# 추가적으로 히트맵을 통해 살펴보기
# 먼저 변수들간의 상관행렬을 생성한다
x_cor = cor(x)

# 생성된 행렬
round(x_cor, 2)
```

```
##             MEI   CO2   CH4   N2O CFC.11 CFC.12   TSI Aerosols  Temp
## MEI        1.00 -0.15 -0.11 -0.16   0.09  -0.04 -0.08     0.35  0.14
## CO2       -0.15  1.00  0.87  0.98   0.40   0.82  0.02    -0.37  0.75
## CH4       -0.11  0.87  1.00  0.89   0.71   0.96  0.15    -0.29  0.70
## N2O       -0.16  0.98  0.89  1.00   0.41   0.84  0.04    -0.35  0.74
## CFC.11     0.09  0.40  0.71  0.41   1.00   0.83  0.28    -0.03  0.38
## CFC.12    -0.04  0.82  0.96  0.84   0.83   1.00  0.19    -0.24  0.69
## TSI       -0.08  0.02  0.15  0.04   0.28   0.19  1.00     0.08  0.18
## Aerosols   0.35 -0.37 -0.29 -0.35  -0.03  -0.24  0.08     1.00 -0.39
## Temp       0.14  0.75  0.70  0.74   0.38   0.69  0.18    -0.39  1.00
```

```
# 히트맵을 그려 상관관계를 살펴본다
ggcorrplot(x_cor, hc.order=TRUE, type='lower', lab=TRUE,
outline.color='white', colors=diverge_hcl(3, palette='Blue Red2'))
```



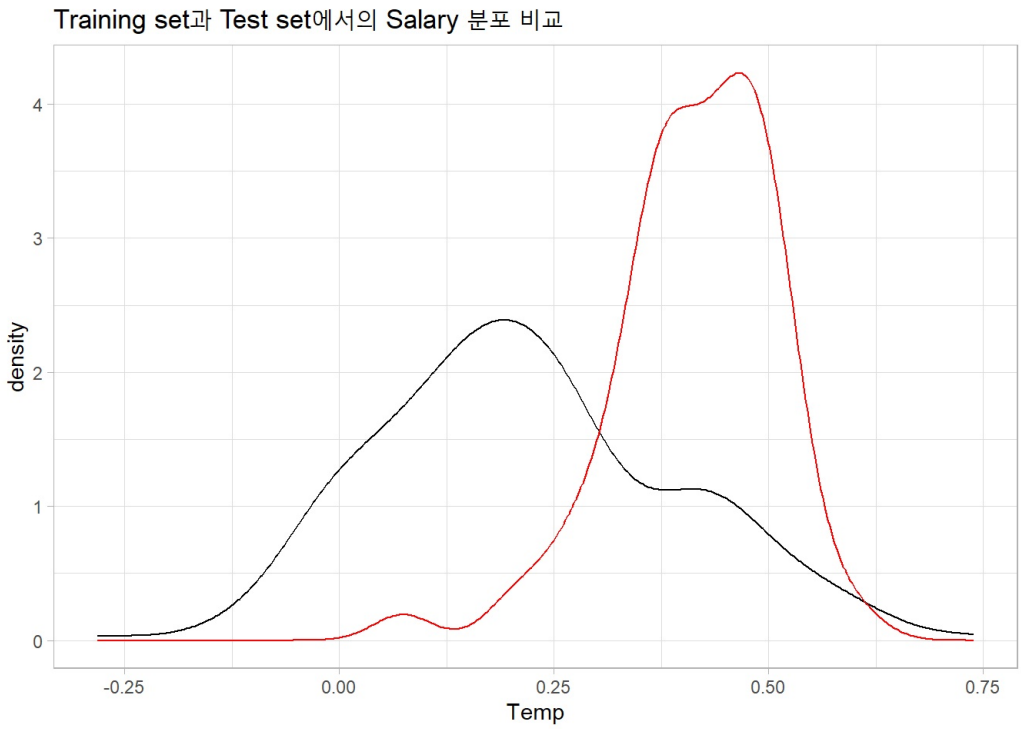
히트맵을 통해 target 변수인 Temp를 제외하고 CH4, CFC.12, N2O, CO2 등의 변수들이 서로 높은 상관관계를 갖음을 확인할 수 있었다. 따라서 이렇게 상관관계가 높은 변수들이 모델에 모두 포함된다면 다중공선성의 문제가 발생할 수 있을것으로 추측된다.

1.2번

2004년 이후의 데이터를 test set으로 2003년까지의 데이터를 training set으로 분할하자. 그리고 training set을 활용하여 linear regression model을 수립하자. 이때 8개의 feature변수를 모두 포함시킨다.

[데이터 전처리]

```
# 문제에서 주어진대로 train set과 test set을 나눈다
# 나눈 후 year와 moth에 해당하는 열은 제외한다
train = subset(clim, Year<=2003)[, -c(1,2)]
test = subset(clim, Year>2003)[, -c(1,2)]

# 나뉜 set의 Temp 분포를 살펴보기 위해 그래프를 그린다
library(ggplot2)
ggplot(data=train, aes(x=Temp)) +
  geom_density() +
  geom_density(data=test, aes(x=Temp), color='red') +
  theme_light() +
  labs(title='Training set과 Test set에서의 Salary 분포 비교')
```



위의 그림에서 검정색은 train set의 분포, 빨간색은 test set의 분포이다. 전체적으로 test set의 분포가 높게 형성되어있음을 확인할 수 있다. 이는 데이터 분포를 고려하지 않고 문제에서 주어진대로 set을 나누었기 때문이다.

[모델 생성]

```
# Temp를 제외한 나머지 8개의 변수를 포함한 linear regression model 생성
model1 = lm(Temp~., data=train)
summary(model1)
```

```
##
## Call:
## lm(formula = Temp ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.25789 -0.06259 -0.00590  0.05567  0.32702
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.385e+02  2.333e+01  -5.939 1.00e-08 ***
## MEI          6.528e-02  6.870e-03   9.501  < 2e-16 ***
## CO2          6.632e-03  2.643e-03   2.509 0.012768 *
## CH4          2.013e-04  5.939e-04   0.339 0.734969
## N2O         -2.525e-02  1.092e-02  -2.312 0.021635 *
## CFC.11      -7.666e-03  1.972e-03  -3.887 0.000131 ***
## CFC.12       4.557e-03  1.267e-03   3.598 0.000390 ***
## TSI          1.051e-01  1.792e-02   5.866 1.48e-08 ***
## Aerosols    -1.590e+00  2.269e-01  -7.010 2.42e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09482 on 239 degrees of freedom
## Multiple R-squared:  0.7133, Adjusted R-squared:  0.7037
## F-statistic: 74.32 on 8 and 239 DF,  p-value: < 2.2e-16
```

1.2(a)번

어떠한 feature들이 Temp에 큰 영향을 미치는가?

```
# 변수와 해당 계수 추출
model1_coef= summary(model1)$coefficients[-1,1]

# 계수의 절대값이 큰 순으로 정렬
model1_coef[order(-abs(model1_coef))]
```

```
##      Aerosols           TSI           MEI           N2O        CFC.11
## -1.5902546637  0.1051169429  0.0652783045 -0.0252485896 -0.0076657488
##           CO2        CFC.12           CH4
##  0.0066321182  0.0045574662  0.0002012985
```

먼저 feature들 중 타겟변수인 Temp에 미치는 영향의 정도를 파악하기 위해 계수를 추출하고 계수의 절대값이 큰 순으로 정렬하였다. 그 결과 변수 Aerosols의 계수가 -1.59 정도로 Temp에 가장 큰 영향을 미치는 것을 확인하였다. 하지만 부호가 음수인 것을 보아 해당 변수가 Aerosols 변수가 1 증가할수록 평균기온(Temp)는 -1.59정도씩 감소한다고 해석할 수 있다.

1.2(b)번

N2O와 CFC-11은 지구의 지표면에서 우주로 발산하는 적외선 복사열을 흡수하여 지구 표면의 온도를 상승시키는 역할을 하는 온실가스로 알려져 있다. 모델에서 N2O와 CFC-11 변수의 coefficient는 양수 값을 가지는가? 음수 값을 가지는가? 만약 음수값을 가진다면 N2O와 CFC-11의 양이 증가할수록 평균 기온이 감소한다는 것을 의미하므로 일반적인 지식과 모순된다. 이러한 모순된 결과가 도출되는 원인은 무엇일까?

```
# 변수와 해당 계수 추출
df1 = summary(model1)$coefficients[-1,1]

# N20 변수의 계수 출력
print(df1['N2O'])
```

```
##         N2O
## -0.02524859
```

```
# CFC.11 변수의 계수 출력
print(df1['CFC.11'])
```

```
##       CFC.11
## -0.007665749
```

두 변수의 출력 결과 두 계수 모두 음수임을 확인하였다. 이는 종속변수 N20와 CFC.11가 증가할수록 평균 기온(Temp)가 감소한다는 것을 의미한다. 이는 일반적인 지식과 모순되며 모델을 통해 잘못된 결과가 도출되었음을 알 수 있다. 이러한 원인으로는 일반적으로 3가지를 생각해볼 수 있다.

먼저, 변수의 한계이다. 모델은 현실을 완벽하게 재현할 수 없기 때문에 현실에서는 존재하나 모델에서는 고려되지 못한 변수가 존재할 수 있다. 이렇게 고려되지 못한 변수로 인해 결과가 모순되는 경우가 있을 수 있다. 두번째로는 측정 오차이다. 주어진 데이터도 수집된 데이터이기 때문에 측정하는 과정에서 오차가 생겨 결과가 모순될 수도 있다.

마지막으로는 다중공선성의 문제이다. 다중공선정은 변수들간의 강한 선형관계가 존재할 경우 발생하는 문제이다. 이렇게 다중공선성이 발생하면 분산이 커져 모델의 안정성이 떨어지며 각 독립변수의 영향력을 제대로 파악하기 어렵다. 따라서 이러한 영향으로 모델의 결과가 제대로 도출되지 않을 수 있다. 주어진 데이터와 같은 경우 1.1번에서 확인했듯이 상관관계가 높은 변수들이 다수 존재한다. 따라서 이러한 모순된 결과는 다중공선성의 문제로 인해 발생했을 것으로 추측된다.

1.3번

MEI, TSI, Aerosols, N2O 4개의 feature만 사용하여 regression model을 만들어 보자.

[모델 생성]

```
# 문제에서 주어진 4개의 변수만와 타겟변수를 추출한다
train_4 = train[c('MEI','TSI','Aerosols','N2O','Temp')]
test_4 = test[c('MEI','TSI','Aerosols','N2O','Temp')]

# 4개의 변수로 모델을 생성한다
model2 = lm(Temp~., data=train_4)
summary(model2)
```

```
##
## Call:
## lm(formula = Temp ~ ., data = train_4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.27840 -0.05990 -0.00633  0.05624  0.34294
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.169e+02  2.315e+01  -5.052 8.58e-07 ***
## MEI          6.492e-02  7.089e-03   9.158  < 2e-16 ***
## TSI          8.003e-02  1.712e-02   4.674 4.89e-06 ***
## Aerosols    -1.708e+00  2.290e-01  -7.459 1.54e-12 ***
## N2O          2.524e-02  1.811e-03  13.933  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09935 on 243 degrees of freedom
## Multiple R-squared:  0.6799, Adjusted R-squared:  0.6747
## F-statistic: 129.1 on 4 and 243 DF,  p-value: < 2.2e-16
```

1.3(a)번

N2O 변수의 coefficient를 2번 모델과 비교해 보자.

```
# 1.2번의 모델에서 변수들의 계수 추출
df1 = summary(model1)$coefficients[-1,]
# 1.3번의 모델에서 변수들의 계수 추출
df2 = summary(model2)$coefficients[-1,]

# 계수 비교를 위해 두 데이터프레임 병합
df_mer = rbind(df1['N2O',], df2['N2O',])
rownames(df_mer) = c("8개(N2O)", "4개(N2O)") # 각 이름 붙여주기
print(df_mer)
```

```
##             Estimate  Std. Error   t value     Pr(>|t|)
## 8개(N2O) -0.02524859 0.010921178 -2.311893 2.163463e-02
## 4개(N2O)  0.02524039 0.001811494 13.933466 7.912925e-33
```

계수를 통해 결과를 해석해보자면, 변수 8개 모델의 N2O같은 경우 해당 변수가 1만큼 증가할수록 평균 기온은 -0.132만큼 감소하며, 변수 4개의 모델의 경우 1만큼 증가할수록 평균 기온은 0.132만큼 증가한다라고 해석할 수 있다. 1.2(b)에서 언급했듯이 변수 8개 모델과 같은 경우 다중공선성의 문제로 일반적인 지식과 모순된 결과로 나왔었지만, 변수를 줄인 결과 일반적인 지식에 맞는 결과를 얻을 수 있었다. 또한 p값을 살펴보면 변수 4개 모델에서의 N2O가 변수 개에서의 N2O보다 더 작은 것을 확인할 수 있으며, 이를 통해 유의하다고 판단할 수 있다.

1.3(b)번

두 모델의 R^2값, Adusted R^2값, test set error (test set에 대한 RMSE) 를 비교해 보자. 어떤 모델이 더 좋은 모델이라고 할 수 있는가?

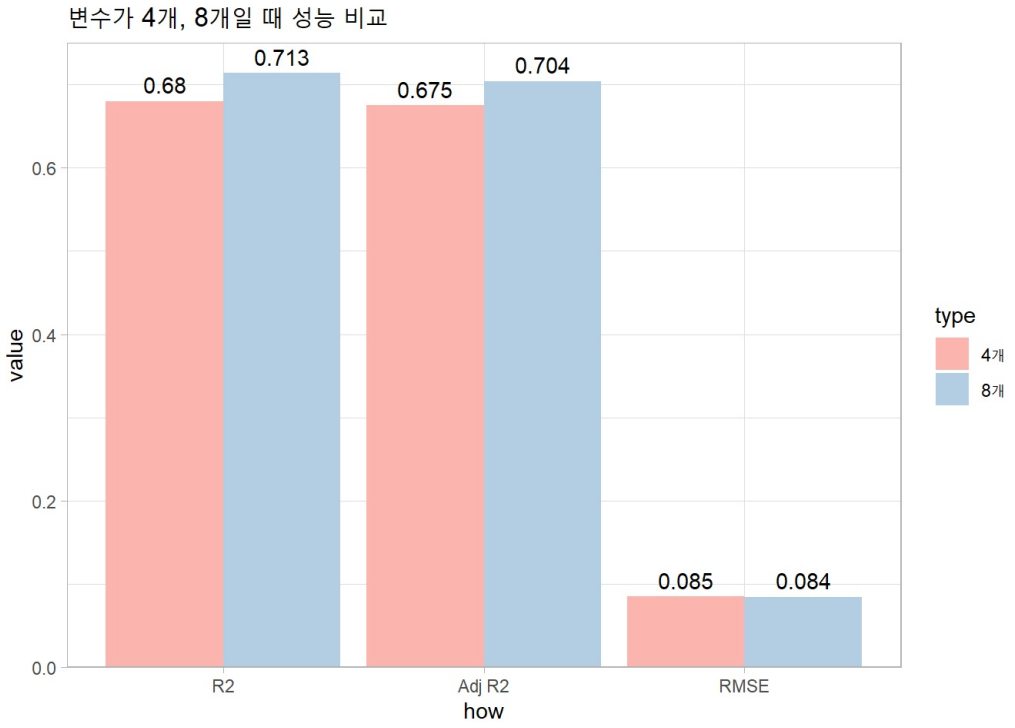
```
# 두 모델 각각의 예측값을 저장
pred1 = predict(model1, test)
pred2 = predict(model2, test_4)

# 두 모델 각각의 R^2값을 저장
r2_1 = summary(model1)$r.squared
r2_2 = summary(model2)$r.squared

# 두 모델 각각의 Adusted R^2값을 저장
adj_1 = summary(model1)$adj.r.squared
adj_2 = summary(model2)$adj.r.squared

# 두 모델 각각의 RMSE값을 저장
rmse1 = sqrt(mean((pred1-test$Temp)^2))
rmse2 = sqrt(mean((pred2-test_4$Temp)^2))

# 각각의 성능지표를 쉽게 비교하기 위해 한 데이터프레임에 저장
perf = data.frame(R2=c(r2_1, r2_2),
                  Adj_R2=c(adj_1, adj_2),
                  RMSE=c(rmse1, rmse2))
rownames(perf) = c("8개", "4개") # 각 행에 맞는 모델이름 저장

# 각 성능 결과를 출력
print(perf)
```

```
##            R2    Adj_R2       RMSE
## 8개 0.7132740 0.7036765 0.08439069
## 4개 0.6799437 0.6746752 0.08501107
```

두 모델에서 구한 각각의 성능은 다음과 같다. 추가적으로 그림을 통해 비교해보도록 하겠다.

[결과 비교]

```
# plot을 그리기 위해 데이터 형태 변환
perf$type = rownames(perf) # 인덱스에 저장했던 정보를 열에 새롭게 추가
# gather함수를 통해 형태 변환
df_gather = gather(perf, key='how', value='value', -type)
df_gather$how = factor(df_gather$how, level=c('R2','Adj_R2','RMSE'),
                       labels=c('R2','Adj R2','RMSE')) # 단순히 출력 순서를 위해

# 각 성능지표를 막대그래프로 표현하여 비교한다
ggplot(data=df_gather, aes(x=how, y=value, group=type, fill=type)) +
  geom_bar(stat='identity', position = 'dodge') +
  theme_light() +
  labs(title='변수가 4개, 8개일 때 성능 비교') +
  scale_y_continuous(expand=c(0,0), limits=c(0, 0.75)) +
  geom_text(aes(label=round(value,3), y=value+0.02), position=position_dodge(width=0.9)) +
  scale_fill_brewer(palette="Pastel1")
```



해당 그림을 통해 두 모델의 R2값, Adj R2값, RMSE 값을 비교해보았다. 그 결과 Adj R2값이 높고, RMSE값이 더 낮은 변수 8개를 사용한 모델의 성능이 더 좋다고 판단된다. 하지만 변수 4개인 모델과 비교했을 때 RMSE값의 차이가 크지 않다. 따라서 변수가 많아 해석하기 보다 어려운 변수 8개 모델보다 더 적은 변수를 사용하면서 비슷한 성능을 내는 변수 4개 모델이 더 합리적이라고 판단된다.

1.4번

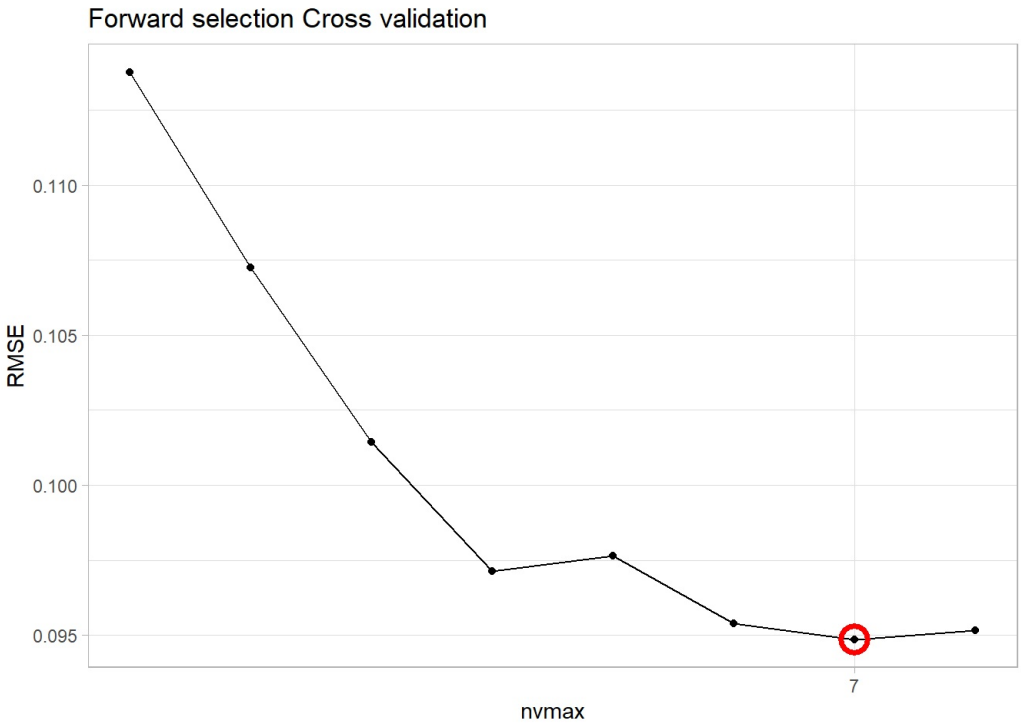
8개의 feature를 대상으로 cross validation을 활용한 stepwise variable selection을 수행해보자.

1.4(a)번

Forward selection과 Backward selection의 결과를 비교해보자.

[Forward selection]

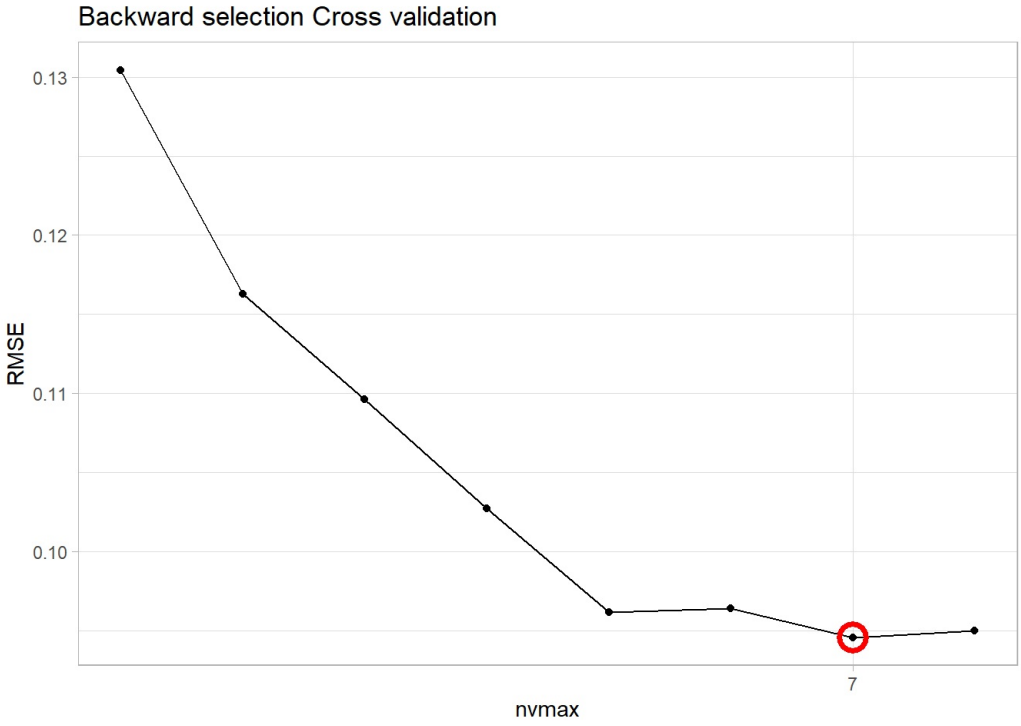
```
set.seed(1)
# cross validation 반복 횟수를 지정한다
train.control = trainControl(method='repeatedcv', number=10, repeats=10)

# forward selection 방법을 사용하여 cross validation을 수행한다
fwd_model = train(Temp~., data=train, method='leapForward',
                  tuneGrid=data.frame(nvmax=1:8), trControl=train.control)

# 모델의 모든 평가지표를 추출한다
result = fwd_model$results

# 평가지표 중 RMSE, R2, MAE값들과 nvmax값만 추출한다
result = result[, c(1:4)]
result_gather = gather(result, key='perform', value='value', -nvmax)

# 최적 nvmax 개수를 저장한다
fwd_best = fwd_model$bestTune[1,]

# 최적 RMSE값을 저장한다
min_fwd = min(fwd_model$result$RMSE)

# 1.6번 문제에서 사용하기 위해 미리 결과값을 저장한다
coef_fwd_cv = coef(fwd_model$finalModel, fwd_model$bestTune$nvmax) # 최적 개수로 모델 생성
pred_fwd = predict(fwd_model, test) # 예측값 저장
fwd_rmse = RMSE(pred_fwd, test$Temp) # 예측값과 실제값의 RMSE값 저장

# nvmax 개수 시각화
ggplot(result, aes(x=nvmax, y=RMSE)) +
  geom_point() +
  geom_line() +
  theme_light() +
  labs(title="Forward selection Cross validation") +
  geom_point(aes(x = fwd_best, y = min_fwd),
             size = 5, color = "red", shape=1, stroke=2) +
  scale_x_continuous(breaks=fwd_best)
```



Forward selection 결과 7개일때 RMSE값이 가장 작은 것을 확인할 수 있다.

[Backward selection]

```
# 모델 생성
bwd_model = train(Temp~., data=train, method='leapBackward',
                  tuneGrid=data.frame(nvmax=1:8), trControl=train.control)
bwd_best = bwd_model$bestTune[1,]
min_bwd = min(bwd_model$result$RMSE)

# 예측 결과값 미리 저장
coef_bwd_cv = coef(bwd_model$finalModel, bwd_model$bestTune$nvmax)
pred_bwd = predict(bwd_model, test)
bwd_rmse = RMSE(pred_bwd, test$Temp)

# nvmax 개수 시각화
ggplot(bwd_model$results, aes(x=nvmax, y=RMSE)) +
  geom_point() +
  geom_line() +
  theme_light() +
  labs(title="Backward selection Cross validation") +
  geom_point(aes(x = bwd_best, y = min_bwd),
             size = 5, color = "red", shape=1, stroke=2) +
  scale_x_continuous(breaks=bwd_best)
```



Backward selection 결과 Forward selection의 결과와 동일하게 7개일때 RMSE값이 가장 작은 것을 확인할 수 있다.

1.4(b)번

Cross validated RMSE가 가장 낮은 best 모델을 결정하자. 어떠한 변수들이 best 모델에 포함되는가?

```
# 각각의 최적 nvmax값과 RMSE값 저장
df_compare = data.frame(nvmax=c(fwd_best, bwd_best),
                        RMSE=c(min_fwd, min_fwd))

# 각 행의 이름 지정
rownames(df_compare) = c("Forward", "Backward")
# 비교 결과 출력
print(df_compare)
```

```
##          nvmax       RMSE
## Forward      7 0.09486619
## Backward     7 0.09486619
```

1.4의 (a)번과 위의 출력 결과를 통해 Forward를 사용한 모델과 Backward 방식을 사용한 두 모델의 최적 변수의 개수와 RMSE값이 동일함을 확인할 수 있다.

```
# 두 모델의 계수 비교
t(cbind(coef_fwd_cv, coef_bwd_cv))
```

```
##             (Intercept)       MEI         CO2         N2O       CFC.11
## coef_fwd_cv   -137.6198 0.0650726 0.006578998 -0.02401335 -0.007564601
## coef_bwd_cv   -137.6198 0.0650726 0.006578998 -0.02401335 -0.007564601
##                  CFC.12       TSI  Aerosols
## coef_fwd_cv 0.004603344 0.1043919 -1.591637
## coef_bwd_cv 0.004603344 0.1043919 -1.591637
```

또한 추출된 변수와 각 계수를 비교해봤을때 모두 동일함을 확인하였다.

[test set의 RMSE값 비교]

```
# 두 모델의 변수의 개수와 RMSE값을 데이터프레임으로 저장한다
df_compare = data.frame(nvmax=c(fwd_best, bwd_best),
                        RMSE=c(fwd_rmse, bwd_rmse))
rownames(df_compare) = c("Forward", "Backward")
print(df_compare)
```

```
##          nvmax       RMSE
## Forward      7 0.08359067
## Backward     7 0.08359067
```

마지막으로 문제에서 요구한 test set에 대한 RMSE값을 각각 구한 결과 모두 같았다. 위에서 보였듯, 추출된 변수와 RMSE값이 모두 동일한 것을 보아 두 모델이 같은 모델임을 알 수 있다. 따라서 두 모델은 동일 모델이므로 성능과 해석 용이성에 차이가 없다. 따라서 해당 모델 자체가 best 모델이다.

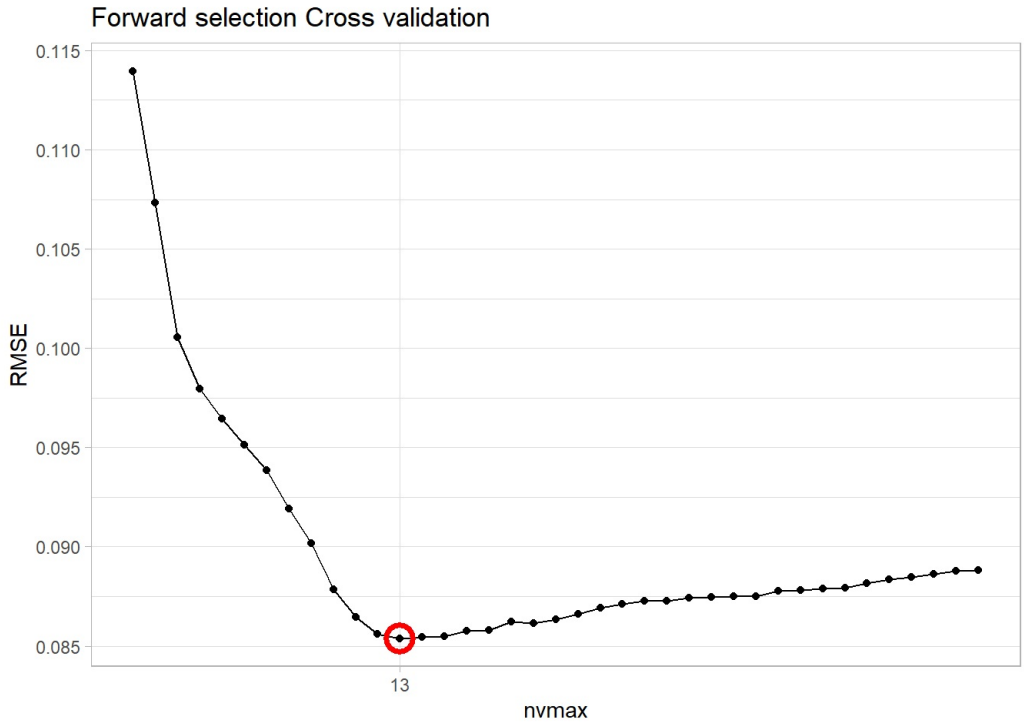
1.5번

Prediction accuracy를 높이기 위해, 기존 8개의 feature들 외에 feature들 사이의 모든 interaction effect, 그리고 CO2, CFC.11, CFC.12의 제곱항들을 모두 추가한 모델을 대상으로 cross validation을 활용한 stepwise variable selection을 수행해보자.

1.5(a)번

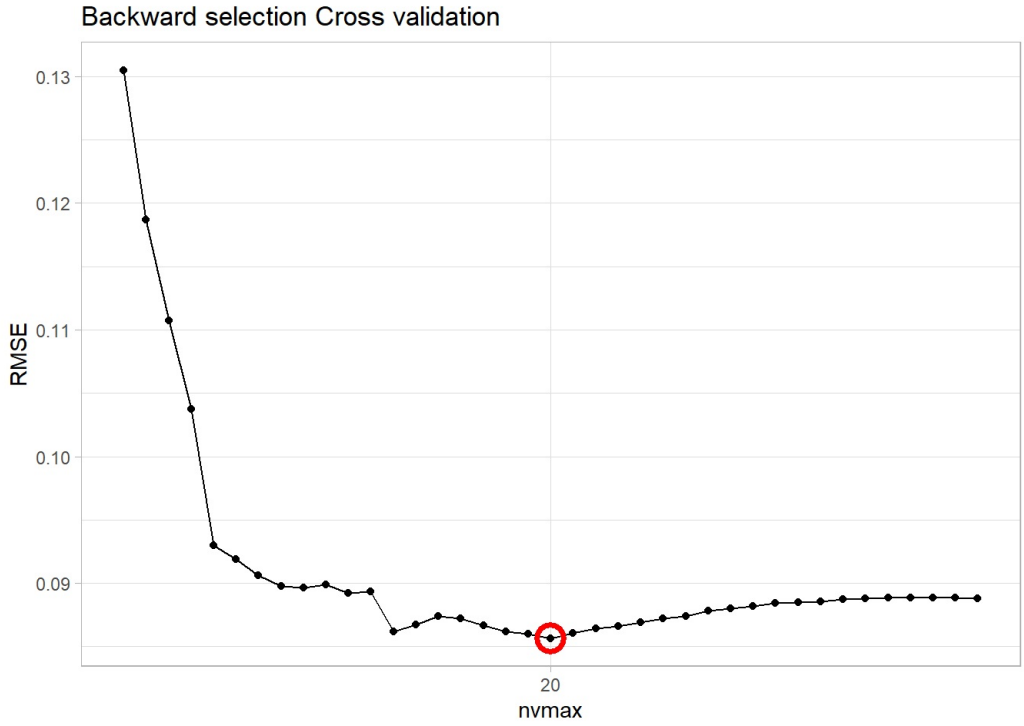
[Forward selection]

```
set.seed(1)
# 모델에 문제에서 주어진 변수 추가
# forward selection 수행
multi_fwd = train(Temp~(.)^2 + I(CO2^2)+I(CFC.11^2)+I(CFC.12^2), data=train, method='leapForward',
                  tuneGrid=data.frame(nvmax=1:39), trControl=train.control)

# best 모델의 정보를 저장한다
multi_best_fwd = multi_fwd$bestTune[1,]
min_result_fwd = min(multi_fwd$results$RMSE)
coef_multi_fwd = coef(multi_fwd$finalModel, multi_fwd$bestTune$nvmax)

# test set의 성능을 저장한다
pred_multi_fwd = predict(multi_fwd, test)
multi_rmse_fwd = RMSE(pred_multi_fwd, test$Temp)

# nvmax 개수 시각화
ggplot(multi_fwd$results, aes(x=nvmax, y=RMSE)) +
  geom_point() +
  geom_line() +
  theme_light() +
  labs(title="Forward selection Cross validation") +
  geom_point(aes(x = multi_best_fwd, y = min_result_fwd),
             size = 5, color = "red", shape=1, stroke=2) +
  scale_x_continuous(breaks=multi_best_fwd)
```



[Backward selection]

```
set.seed(1)
# 모델에 문제에서 주어진 변수 추가
# backward selection 수행
multi_bwd = train(Temp~(.)^2 + I(CO2^2)+I(CFC.11^2)+I(CFC.12^2), data=train, method='leapBackward',
                  tuneGrid=data.frame(nvmax=1:39), trControl=train.control)

# 위와 동일하게 변수 저장
multi_best_bwd = multi_bwd$bestTune[1,]
min_result_bwd = min(multi_bwd$results$RMSE)
coef_multi_bwd = coef(multi_bwd$finalModel, multi_bwd$bestTune$nvmax)

pred_multi_bwd = predict(multi_bwd, test)
multi_rmse_bwd = RMSE(pred_multi_bwd, test$Temp)

ggplot(multi_bwd$results, aes(x=nvmax, y=RMSE)) +
  geom_point() +
  geom_line() +
  theme_light() +
  labs(title="Backward selection Cross validation") +
  geom_point(aes(x = multi_best_bwd, y = min_result_bwd),
             size = 5, color = "red", shape=1, stroke=2) +
  scale_x_continuous(breaks=multi_best_bwd)
```



1.5(b)번

```
# 두 모델의 변수의 수와 RMSE값 추출
df_compare = data.frame(nvmax=c(multi_best_fwd, multi_best_bwd),
                        RMSE=c(min_result_fwd, min_result_bwd))
rownames(df_compare) = c("Forward", "Backward")
print(df_compare)
```

```
##          nvmax       RMSE
## Forward     13 0.08538706
## Backward    20 0.08566027
```

먼저, Forward 방식을 사용한 모델의 최적 변수 개수는 13개이며, RMSE값은 0.085이다. 다음으로 Backward 방식을 사용한 모델의 최적 변수 개수는 20개이며, RMSE값은 0.086이다. 따라서 문제에서 요구한 조건('Cross validated RMSE가 가장 낮은) 부합하는 모델은 Forward selection 방식을 사용한 모델이다.

```
# best 모델(forward selection model)의 계수 출력
coef_multi_fwd[-1][order(-abs(coef_multi_fwd[-1]))]
```

```
## CFC.11:Aerosols CFC.12:Aerosols    CO2:Aerosols             TSI    CH4:Aerosols
##    6.975439e-01   -4.180642e-01    1.245920e-01   -4.712490e-02   -1.338785e-02
##      MEI:CFC.11         MEI:CO2        I(CO2^2)      CO2:CFC.12         CO2:TSI
##    2.290603e-03   -1.476024e-03   -4.258315e-04    1.816509e-04    1.614155e-04
##      N2O:CFC.11     I(CFC.12^2)   CFC.11:CFC.12
##   -8.962274e-05   -6.386884e-05    2.085690e-05
```

best 모델(forward selection model)의 계수를 절댓값이 큰 순으로 정렬한 출력은 위와 같다. 해당 결과를 통해 CFC.11:Aerosols부터 CFC.11:CFC.12까지 총 12개의 변수가 포함된 것을 확인할 수 있다.

1.6번

```
# 각 문제의 모델의 변수의 개수와 RMSE값 저장
df_all_compare = data.frame(nvmax=c(8, 4, 7, 13),
                        RMSE=c(rmse1, rmse2, fwd_rmse, multi_rmse_fwd))
rownames(df_all_compare) = c('2번', '3번', '4번', '5번')
print(df_all_compare)
```

```
##     nvmax       RMSE
## 2번     8 0.08439069
## 3번     4 0.08501107
## 4번     7 0.08359067
## 5번    13 0.09242062
```

문제에서 구한 4개의 모델의 RMSE값들을 비교해본 결과 4번에서 구한 모델의 RMSE값이 0.0839 정도로 가장 낮은 것을 알 수 있다. 5번 모델의 RMSE값과 같은 경우 다른 모델에 비해 성능이 확연히 떨어진다. 이는 너무 많은 변수를 생성하고 포함시켰기 때문이라고 추측된다. 하지만 변수 8개 모델과 4개의 모델을 비교하였을 때 상관관계가 높은 변수들을 모두 포함한 8개의 변수가 더 성능이 잘 나온 것을 알 수 있다. 하지만 이 4개의 변수는 문제에 주어진대로 임의로 설정하였기 때문에 4개의 변수가 타겟변수에 대해 정말 가장 유의한 변수인지 판단할 수 없기 때문이다. 따라서 변수 8개중 유의한 7개의 변수들로 선택된 모델인 4번의 성능이 가장 높은 것은 타당하다.

2. Regression on simulated Data

먼저 아래와 같이 랜덤으로 데이터를 생성하자.

(i)번

rnorm() 함수를 활용해서 평균이 0, 표준편차가 1인 표준정규분포로부터 크기가 100인 vector X를 생성하고, 평균이 0, 표준편차가 4인 정규분포로부터 크기가 100인 오차 vector ϵ을 생성한다. X와 ϵ을 생성하기 위한 rnorm() 함수에 대해서 동일한 random seed 값을 사용하지 않도록 주의하자.

```
# random seed 값을 다르게 부여한다
# 문제에 주어진 조건대로 두 벡터를 생성한다
set.seed(11); x = rnorm(100, mean=0, sd=1)
set.seed(22); ep = rnorm(100, mean=0, sd=4)
```

```
# 생성된 x벡터의 예시
print(head(x, 5))
```

```
## [1] -0.59103110  0.02659437 -1.51655310 -1.36265335  1.17848916
```

```
# 생성된 ϵ 벡터의 예시
print(head(ep, 5))
```

```
## [1] -2.0485564  9.9407347  4.0313046  1.1712583 -0.8358374
```

(ii)번

크기가 인 target vector Y를 다음 식을 사용하여 생성한다.

```
# 문제에서 주어진 수식에 맞게 함수를 생성한다
target_y = function(x,y){
  return (1-2*x+x^2-4*x^3+y)
}

# 함수를 적용한 새로운 y 벡터를 생성한다
y = target_y(x, ep)
print(head(y, 5))
```

```
## [1]  1.308654 10.888178 24.316228 16.874220 -7.350895
```

2.1번

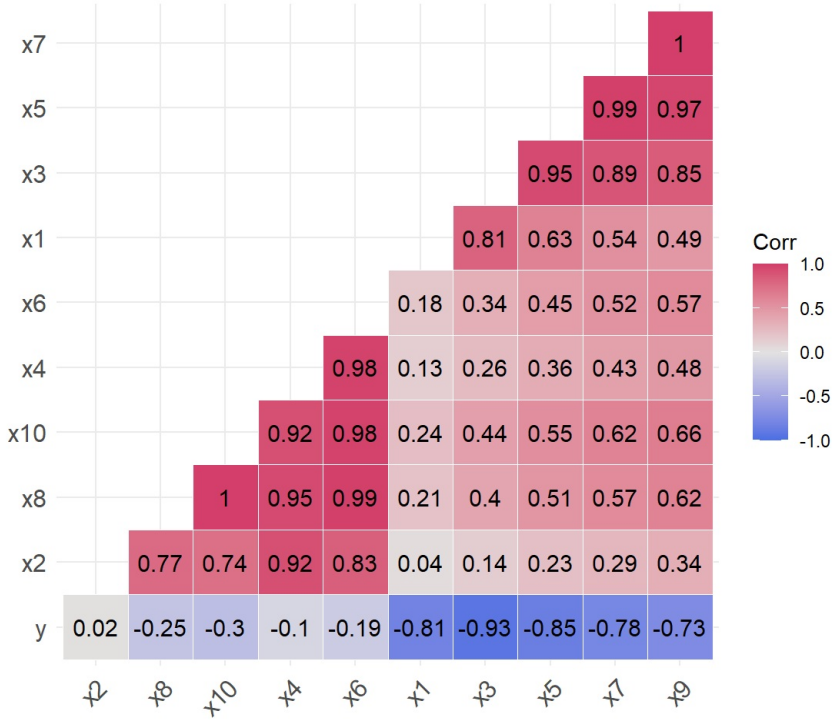
10개 변수를 feature로, Y를 target으로 설정하자. 이때 feature 변수들과 target 변수 사이의 상관관계를 시각화해보자.

```
# for문을 통해 x~x^10까지 생성한다
lst = list() # 결과를 저장할 빈 리스트 생성
for (i in 1:10) {
  lst[[i]] = x^i
}

# 반복문을 통해 생성된 리스트를 데이터프레임 형태로 저장
df = data.frame(x1=lst[[1]],x2=lst[[2]],x3=lst[[3]],x4=lst[[4]],x5=lst[[5]],
                x6=lst[[6]],x7=lst[[7]],x8=lst[[8]],x9=lst[[9]],x10=lst[[10]])

# 생성한 변수들과 y값 데이터프레임 형태로 저장
df_y = cbind(y, df)
str(df_y)
```

```
## 'data.frame':    100 obs. of  11 variables:
##  $ y  : num  1.31 10.89 24.32 16.87 -7.35 ...
##  $ x1 : num  -0.591 0.0266 -1.5166 -1.3627 1.1785 ...
##  $ x2 : num  0.349318 0.000707 2.299933 1.856824 1.388837 ...
##  $ x3 : num  -2.06e-01 1.88e-05 -3.49 -2.53 1.64 ...
##  $ x4 : num  1.22e-01 5.00e-07 5.29 3.45 1.93 ...
##  $ x5 : num  -7.21e-02 1.33e-08 -8.02 -4.70 2.27 ...
##  $ x6 : num  4.26e-02 3.54e-10 1.22e+01 6.40 2.68 ...
##  $ x7 : num  -2.52e-02 9.41e-12 -1.85e+01 -8.72 3.16 ...
##  $ x8 : num  1.49e-02 2.50e-13 2.80e+01 1.19e+01 3.72 ...
##  $ x9 : num  -8.80e-03 6.65e-15 -4.24e+01 -1.62e+01 4.38 ...
##  $ x10: num  5.20e-03 1.77e-16 6.44e+01 2.21e+01 5.17 ...
```

[히트맵]

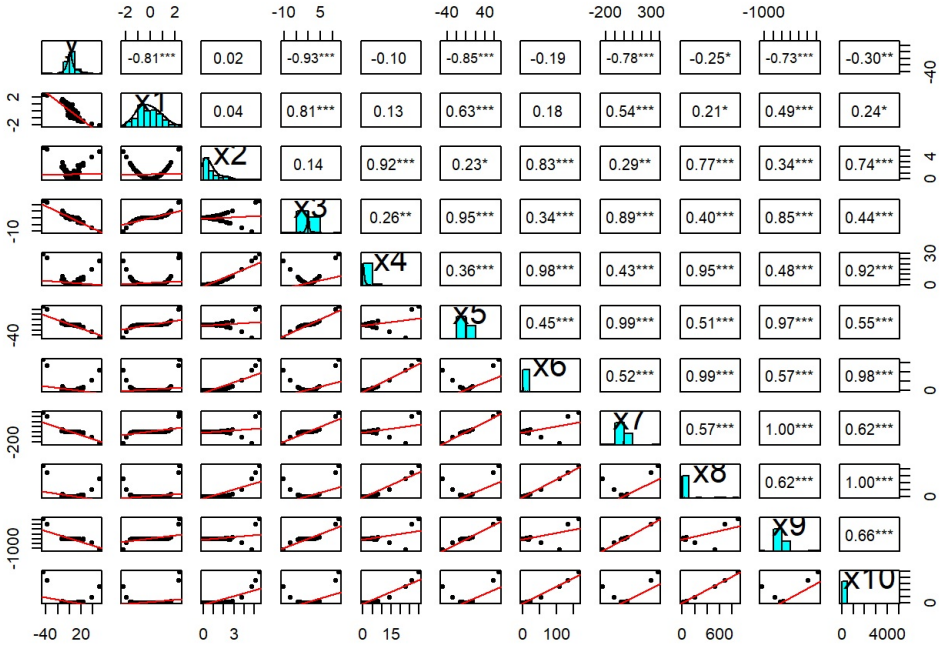
```
# 히트맵을 통해 상관관계 확인
x_cor = cor(df_y)
ggcorrplot(x_cor,
           hc.order=TRUE,
           type='lower',
           lab=TRUE,
           outline.color='white',
           colors=diverge_hcl(3, palette='Blue Red2'))
```



히트맵을 통해 target 변수와 feature 변수들은 모두 음의 상관관계를 갖는다는 것을 확인할 수 있다. 또한 x3, x5, x1, x7, x9 순으로 타겟변수와 높은 음의 상관관계를 갖음을 확인할 수 있다. 이는 해당 변수들이 증가할수록 타겟변수가 감소한다는 것을 의미한다. 추가적으로 분포의 그림을 통해 살펴보도록 하겠다.

[변수들간의 분포]

```
# 전체적인 분포와 상관관계 확인
pairs.panels(df_y, lm=TRUE, ellipses=FALSE, rug=FALSE, stars=TRUE)
```



타겟변수와의 상관관계를 확인하기 위해 첫번째 열을 살펴보도록한다. 히트맵에서 확인한 것과 같이 모든 변수가 x축이 증가할수록 y축이 감소하는 음의 상관관계를 갖음을 확인할 수 있다. 그림 속 상관계수 옆 * 표시를 통해 히트맵에서 높은 상관관계를 갖는 변수들과 모두 유의미한 상관관계를 갖음을 확인할 수 있다. 하지만 낮은 상관관계를 갖는 x1 변수와는 상관관계가 유의하지 않다는 것을 알 수 있다. 마지막으로 y타겟과 다른 변수들의 분포를 살펴보면 우함수와는 대체로 상관관계가 높지 않고 대부분의 기함수와의 상관관계가 높은 것을 알 수 있다. 이는 똑같은 그래프의 형태가 좌우 대칭인 우함수에 비해 원점 대칭 형태인 기함수의 형태가 선형에 더 가깝기 때문이라고 추측할 수 있다.

2.2번

10개의 feature를 모두 포함하는 linear regression model을 만들어보자. 통계적으로 유의한 변수가 있는가? regression coefficient 값을 실제 값과 비교해보자.

[유의한 변수 확인]

```
model2_2 = lm(y~., data=df_y)
summary(model2_2)
```

```
##
## Call:
## lm(formula = y ~ ., data = df_y)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9609 -2.5448 -0.3246  1.9097 12.9845
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.53789    0.83292   3.047  0.00304 **
## x1           -1.07807    2.42238  -0.445  0.65737
## x2          -11.70150    5.55895  -2.105  0.03811 *
## x3           -5.50451    6.30816  -0.873  0.38523
## x4           19.28009    9.32837   2.067  0.04166 *
## x5            0.96085    4.96786   0.193  0.84708
## x6          -10.08686    5.62452  -1.793  0.07631 .
## x7           -0.18108    1.49261  -0.121  0.90371
## x8            2.17644    1.36849   1.590  0.11529
## x9            0.01191    0.14922   0.080  0.93657
## x10          -0.16494    0.11584  -1.424  0.15799
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.848 on 89 degrees of freedom
## Multiple R-squared:  0.8983, Adjusted R-squared:  0.8869
## F-statistic: 78.65 on 10 and 89 DF,  p-value: < 2.2e-16
```

```
# p값이 작은 순으로 나열
pval = summary(model2_2)$coefficients[-1,4]
sort(pval)
```

```
##         x2         x4         x6         x8        x10         x3         x1
## 0.03811085 0.04165597 0.07631011 0.11529288 0.15798662 0.38522789 0.65736628
##         x5         x7         x9
## 0.84707635 0.90371409 0.93656759
```

linear regression의 결과를 통해 도출된 각 변수의 p값을 작은순으로 나열한 결과는 위와 같다. 변수 x2가 통계적으로 가장 유의하며, x4가 다음으로 유의함을 확인할 수 있다. 하지만 변수 x6부터는 p값이 0.05보다 크기 때문에 통계적으로 유의하다고 할 수 없다. 따라서 해당 모델에서 총 2개의 변수가 유의하다고 판단된다.

[계수 비교]

```
# 추정된 계수 추출
coef2_2 = summary(model2_2)$coefficients[-1, 1]
round(coef2_2, 3)
```

```
##      x1      x2      x3      x4      x5      x6      x7      x8      x9     x10
##  -1.078 -11.702  -5.505  19.280   0.961 -10.087  -0.181   2.176   0.012  -0.165
```

linear regression model을 통해 추출한 추정 계수는 위와 같다. 먼저 추정된 x1과 x3은 실제 계수와 같은 부호를 갖고 있으며 값의 차이도 많이 나지 않은 것을 확인할 수 있다. 하지만 x2와 같은 경우 -11로 실제 3임을 고려하면 차이가 많이 난다는 것을 알 수 있다. 또한 실제 계수는 x4이후로 모두 0이라고 할 수 있지만, 추정된 회귀계수 중 x4, x6, x8의 절댓값은 다른 변수들과 비교하더라도 큰 값이다. 따라 10개의 변수를 모두 포함하는 linear regession model을 만들 경우 실제 계수와 차이가 많이 난다는 것을 알 수 있다.

2.3번

X, X2, X3의 3개의 변수를 feature로, Y를 target으로 linear regression model을 만들어보자. 모든 feature들이 통계적으로 유의한가? regression coefficient 값을 실제 계수값과 비교해보자.

```
# 문제에서 주어진 변수만 추출
df2_3 = df_y[, c(1:4)]
str(df2_3)
```

```
## 'data.frame':    100 obs. of  4 variables:
##  $ y : num  1.31 10.89 24.32 16.87 -7.35 ...
##  $ x1: num  -0.591 0.0266 -1.5166 -1.3627 1.1785 ...
##  $ x2: num  0.349318 0.000707 2.299933 1.856824 1.388837 ...
##  $ x3: num  -2.06e-01 1.88e-05 -3.49 -2.53 1.64 ...
```

```
# 모델 생성
model2_3 = lm(y~., data=df2_3)
summary(model2_3)
```

```
##
## Call:
## lm(formula = y ~ ., data = df2_3)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.365 -2.976 -0.387  2.192 13.119
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.9515     0.4934   1.928   0.0568 .
## x1           -1.6829     0.7317  -2.300   0.0236 *
## x2            1.5182     0.3638   4.173  6.6e-05 ***
## x3           -3.7737     0.2655 -14.214  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.856 on 96 degrees of freedom
## Multiple R-squared:  0.8899, Adjusted R-squared:  0.8864
## F-statistic: 258.6 on 3 and 96 DF,  p-value: < 2.2e-16
```

[유의한 변수 확인]

```
# p값 추출
pval = summary(model2_3)$coefficients[-1,4]
sort(round(pval,3))
```

```
##    x2    x3    x1
## 0.000 0.000 0.024
```

해당 모델의 각 변수의 p값을 추출한 결과 모두 0.05보다 작으며 통계적으로 유의함을 알 수 있다.

[계수 비교]

```
# 계수 추출
coef2_3 = summary(model2_3)$coefficients[-1, 1]
round(coef2_3, 3)
```

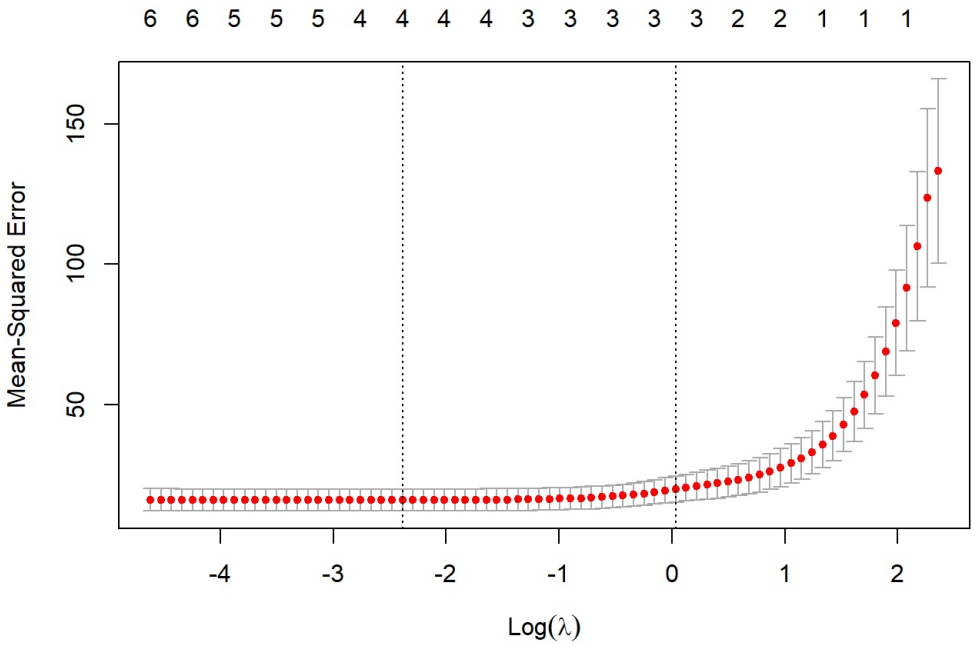
```
##     x1     x2     x3
## -1.683  1.518 -3.774
```

나머지 변수를 넣지 않고 3개의 변수만을 넣었을때 10개의 변수를 모두 넣었을 때보다 계수가 더 비슷해진 것을 확인할 수 있다. 특히 x2와 같은 경우 부호가 다르고 절대값의 차이가 많이 났던 위의 경우와 다르게 부호가 같으며 절대값 차이가 많이 줄어든 것을 확인할 수 있다. 이는 다른 변수들이 제거됨으로써 해당 변수의 영향력을 더욱 잘 반영한 것으로 추측된다.

2.4번

10개의 feature로, Y를 target으로 Lasso regression model을 만들어 본다. cross validation을 통해 합리적인 모델을 찾아보자. 이 모델에는 어떤 변수가 포함되었는가? regression coefficient값을 실제 계수값과 비교해보자. 그리고 결과를 바탕으로 Lasso regression의 효과에 대해 설명해보자.

```
# lasso에대한 cross validation 수행
cv_lasso = cv.glmnet(x=as.matrix(df), y=y, alpha=1, nfolds=10)
plot(cv_lasso)
```




```
# cv error가 최소가 되는 모델에서의 변수와 해당 계수 추출
predict(cv_lasso, s=cv_lasso$lambda.min, type='coefficients')
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  1.11270419
## x1          -1.63910900
## x2           1.22567947
## x3          -3.75675818
## x4           0.04898437
## x5           .
## x6           .
## x7           .
## x8           .
## x9           .
## x10          .
```

```
# 위의 모델보다 단순한 모델에서의 변수와 해당 계수 추출
predict(cv_lasso, s=cv_lasso$lambda.1se, type='coefficients')
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  1.9381050
## x1          -1.3370746
## x2           0.4273868
## x3          -3.4008769
## x4           .
## x5           .
## x6           .
## x7           .
## x8           .
## x9           .
## x10          .
```

위의 결과를 통해 cv error가 최소가 되는 모델의 변수는 x1, x2, x3, x4로 총 4개가 선택되었으며, 'lambda.1se'를 기준으로 선택된 모델의 변수는 x1, x2,x3로 총 3개인 것을 확인하였다. 이런 경우에는 One-Standard-Error-Rule에 따라 cv error가 최소가 되는 모델보다는 cv error가 조금 더 높더라도 단순한 모델을 선택하는 것이 더 합리적일 수 있다. 따라서 문제에서 요구하는 가장 합리적인 모델은두번째 모델이며, 포함된 변수는 x1, x2, x3이다.

해당 모델을 통해 추정된 계수를 실제 계수와 비교해보면 x1과 x3은 -1.33과 -3.4로 실제 계수값과 유사하지만, x2와 같은 경우 0.427정도로 실제 계수인 3과 비교했을때 다른 변수에 비해 차이가 있다고 판단된다.

이렇게 lasso regression은 2.3번과 같이 변수를 직접 선택하지 않더라도 모든 변수들 중 중요한 변수들만을 선택하여 모델링을 해준다. 이처럼 lasso는 변수가 많은 데이터셋에서 모델에 불필요한 변수들을 자동으로 제거하여 모델의 복잡성을 줄이고 해석력을 향상시켜준다.