HW 05

2021234640 이종현

Time series 그리기

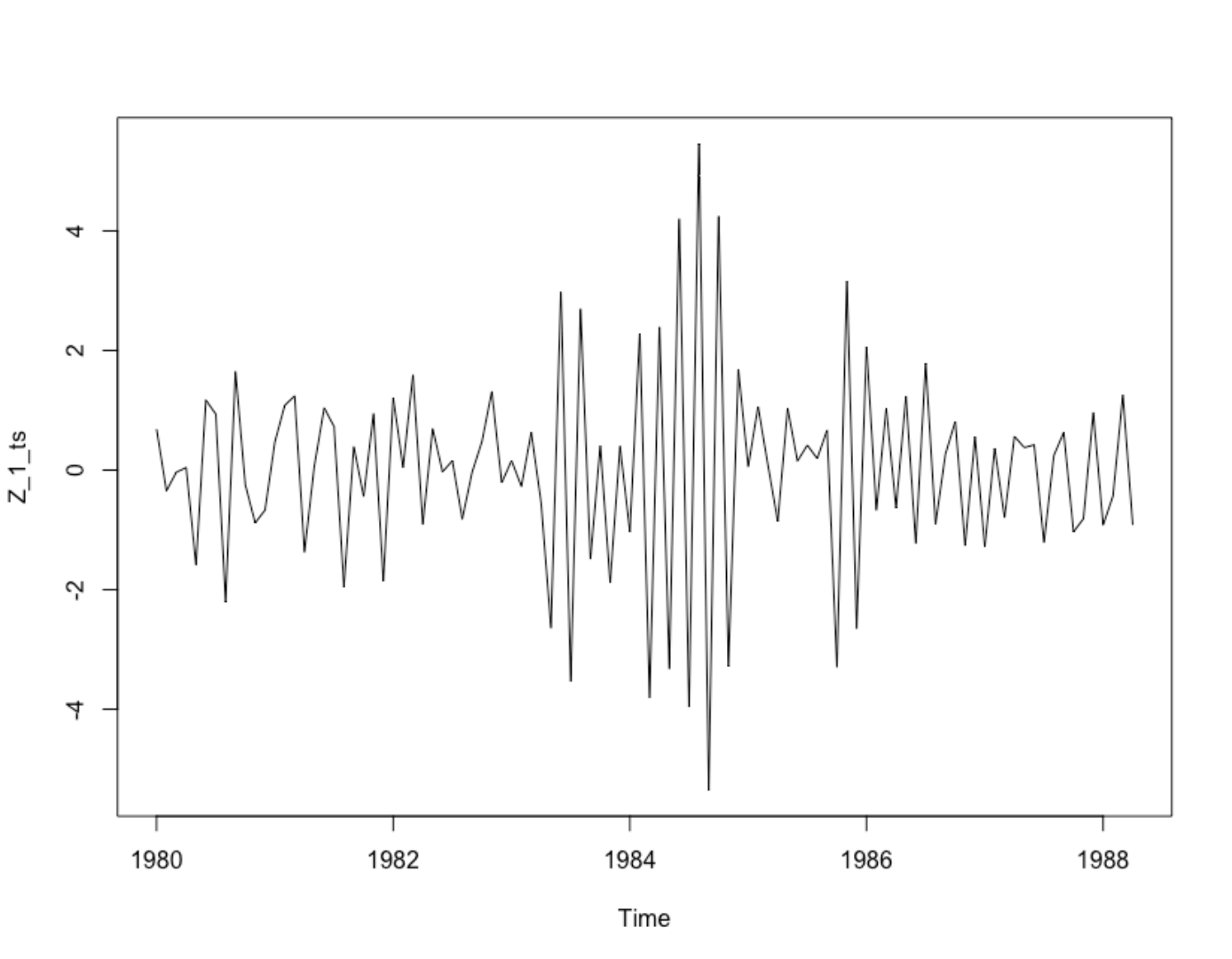
두 시계열은 다음 코드를 이용하여 생성되었다.

```
rm(list=ls())

library(ggplot2)

t = 1:100
Z_1 = c()
z_1_n_1 = 0

Z_2 = c()
z_2_n_1 = 0

for (x in t){
  z_1_n = -0.7 * z_1_n_1 + rnorm(1)
  z_2_n = 0.5 * z_2_n_1 + rnorm(1)

  Z_1 = c(Z_1, z_1_n)
  Z_2 = c(Z_2, z_2_n)

  z_1_n_1 = z_1_n
  z_2_n_1 = z_2_n

}
```

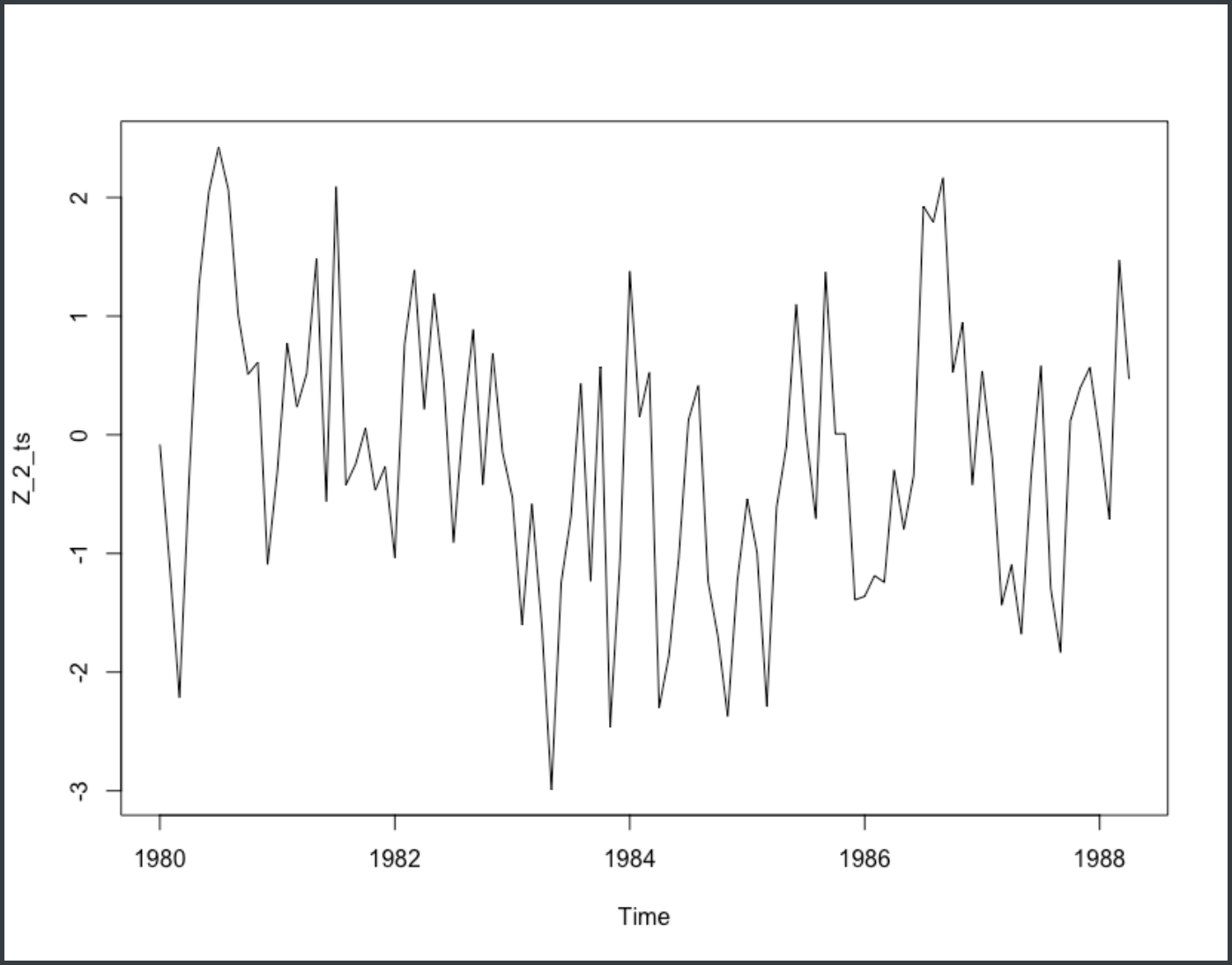
```
Z_1_ts = ts(Z_1, start=c(1980, 1), frequency=12)
Z_2_ts = ts(Z_2, start=c(1980, 1), frequency=12)

ts.plot(Z_1_ts)
ts.plot(Z_2_ts)
```

이때 각 시계열 플랏은 다음과 같다.

- 시계열 1번
  - 다소 wavelet 같은 모양의 시계열이 생성되었다.



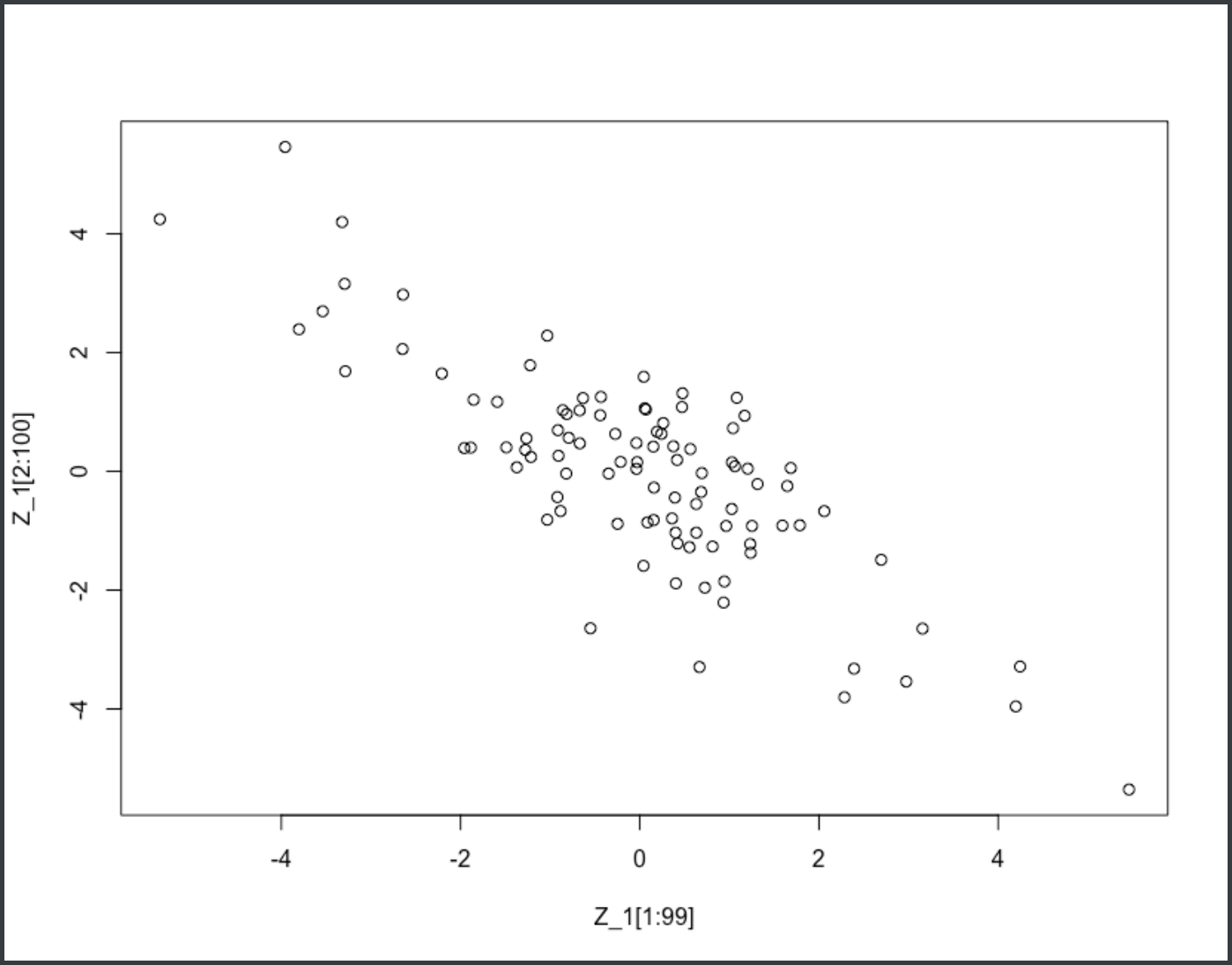
- 시계열 2번



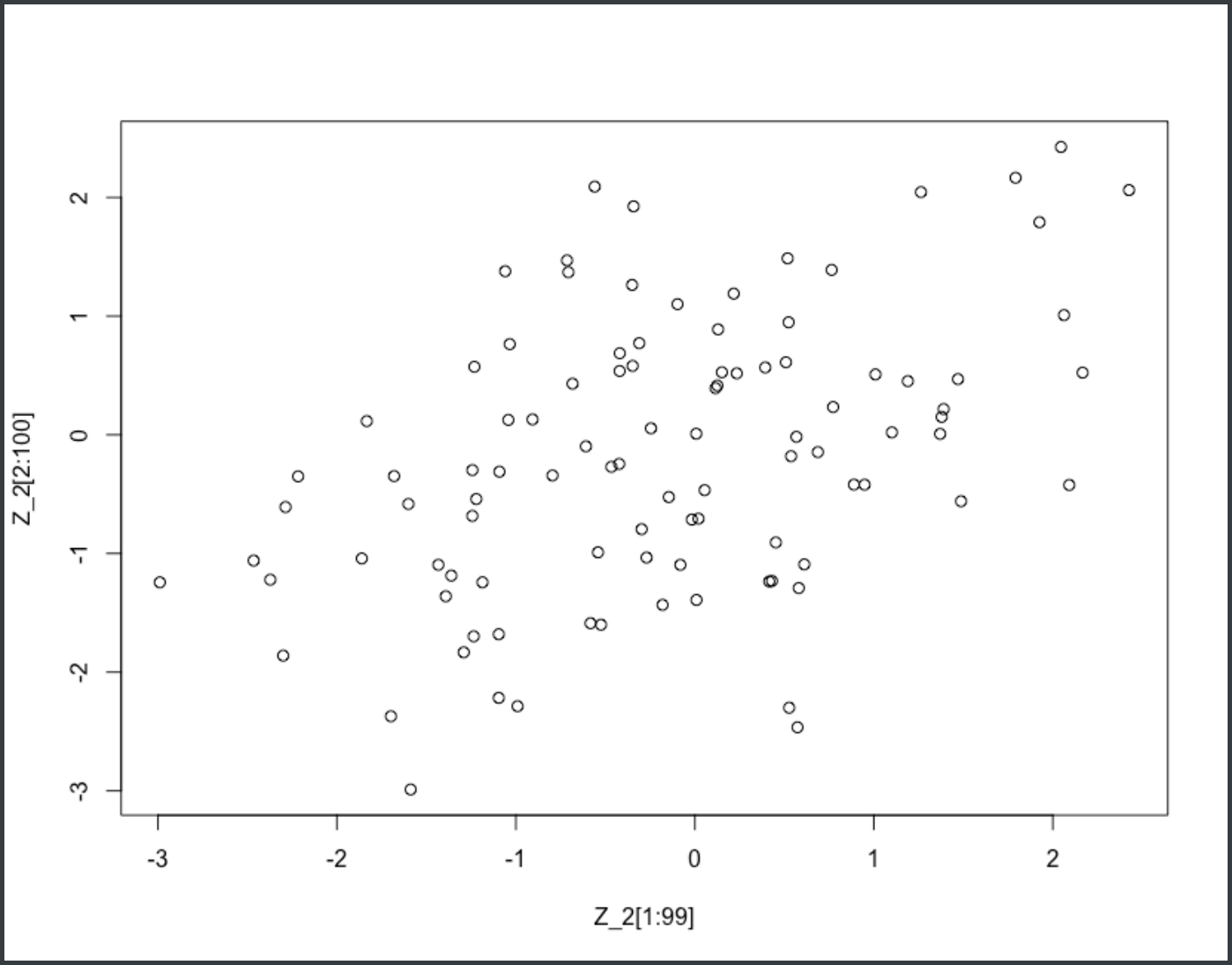
t 시점과 t-1 시점 scatter plot

두 시계열은 자기 상관 시계열로 높은 상관성을 보일 것으로 기대된다.

- 시계열 1번
  - 강한 음의 상관 관계가 보인다. 이는 시계열 생성 당시, coefficient를 -0.7로 주었기 때문이다.
  - 실제 correlation coefficient 도 -0.810807 로 확인되었다.

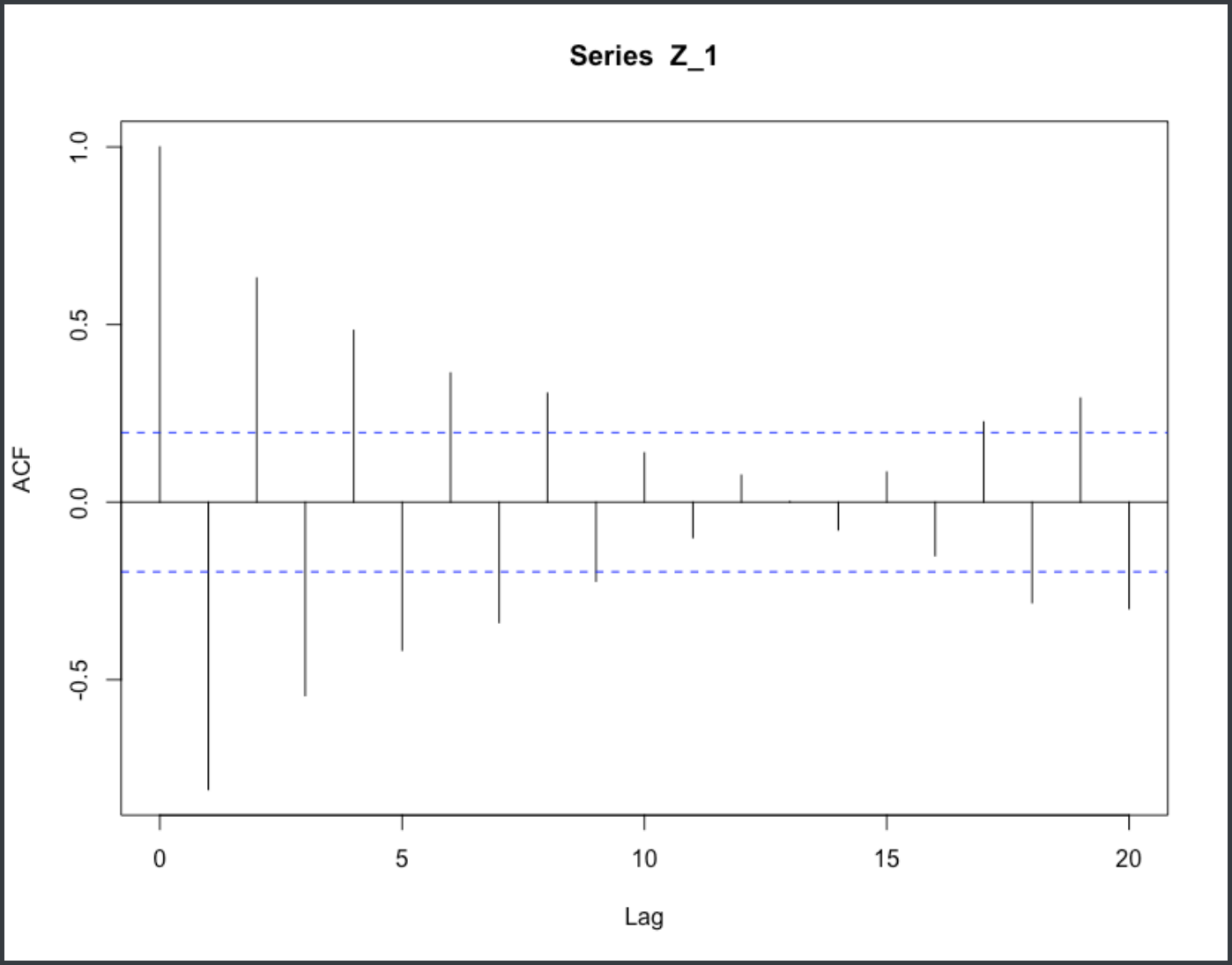


- 시계열 2번
  - 0.5 의 coeffcient 를 바탕으로 생성되었기 때문에 양의 상관관계가 발견된다.
  - 0.5283011 의 상관 계수를 확인할 수 있었다.

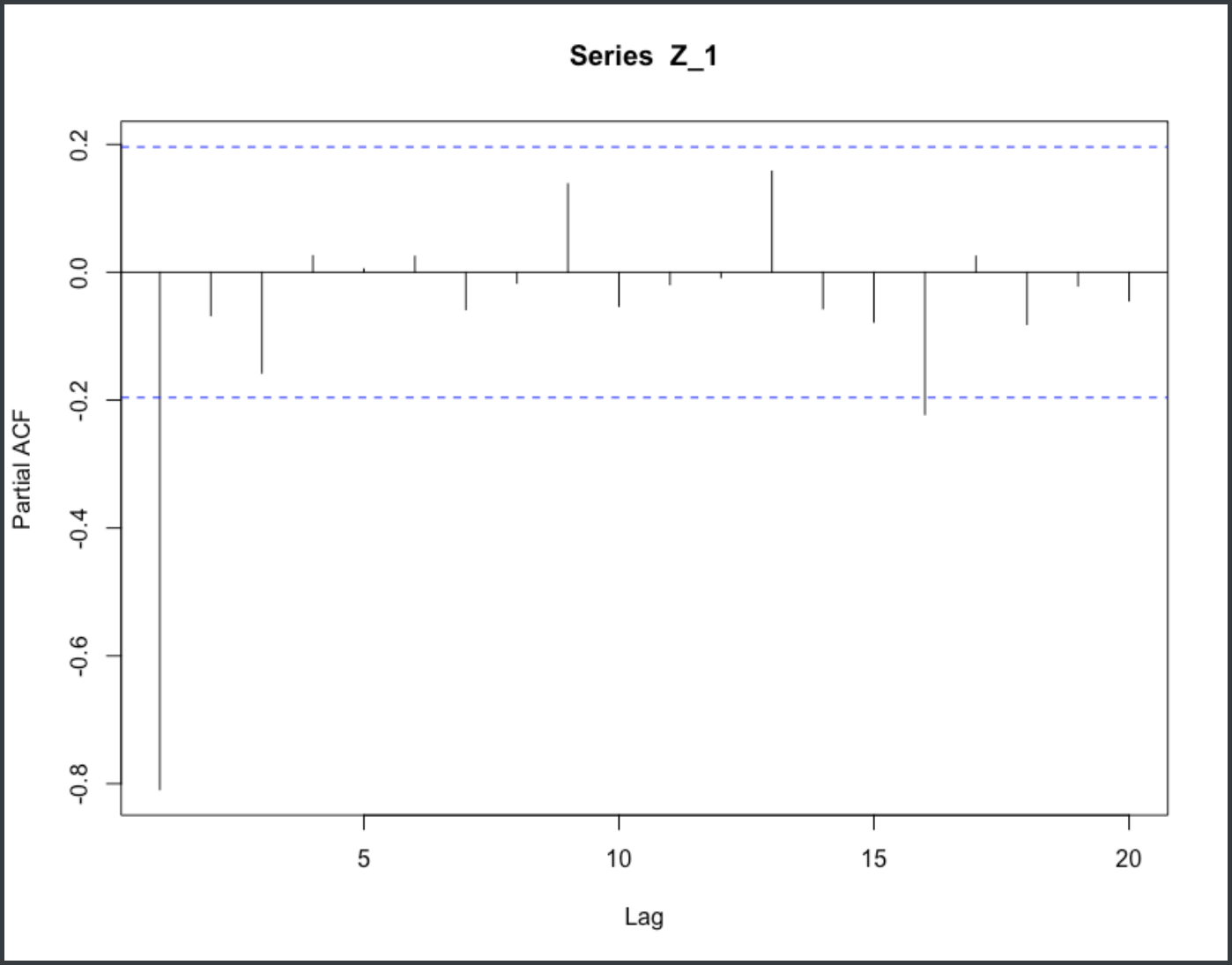


두 시계열의 ACF, PACF

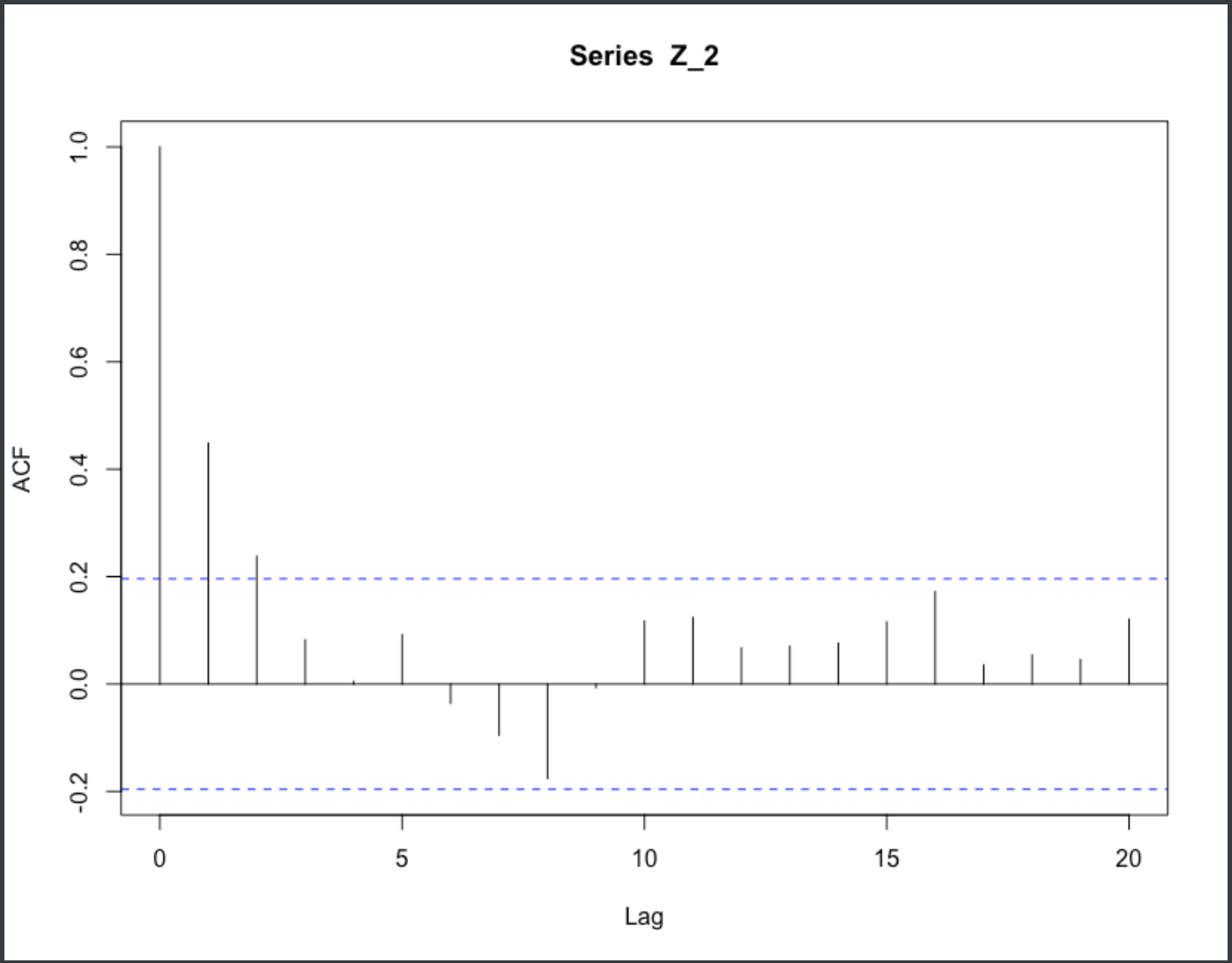
- 시계열 1번
  - ACF 함수는 3번째까지 높은 수준을 유지하고 있음을 확인할 수 있다.



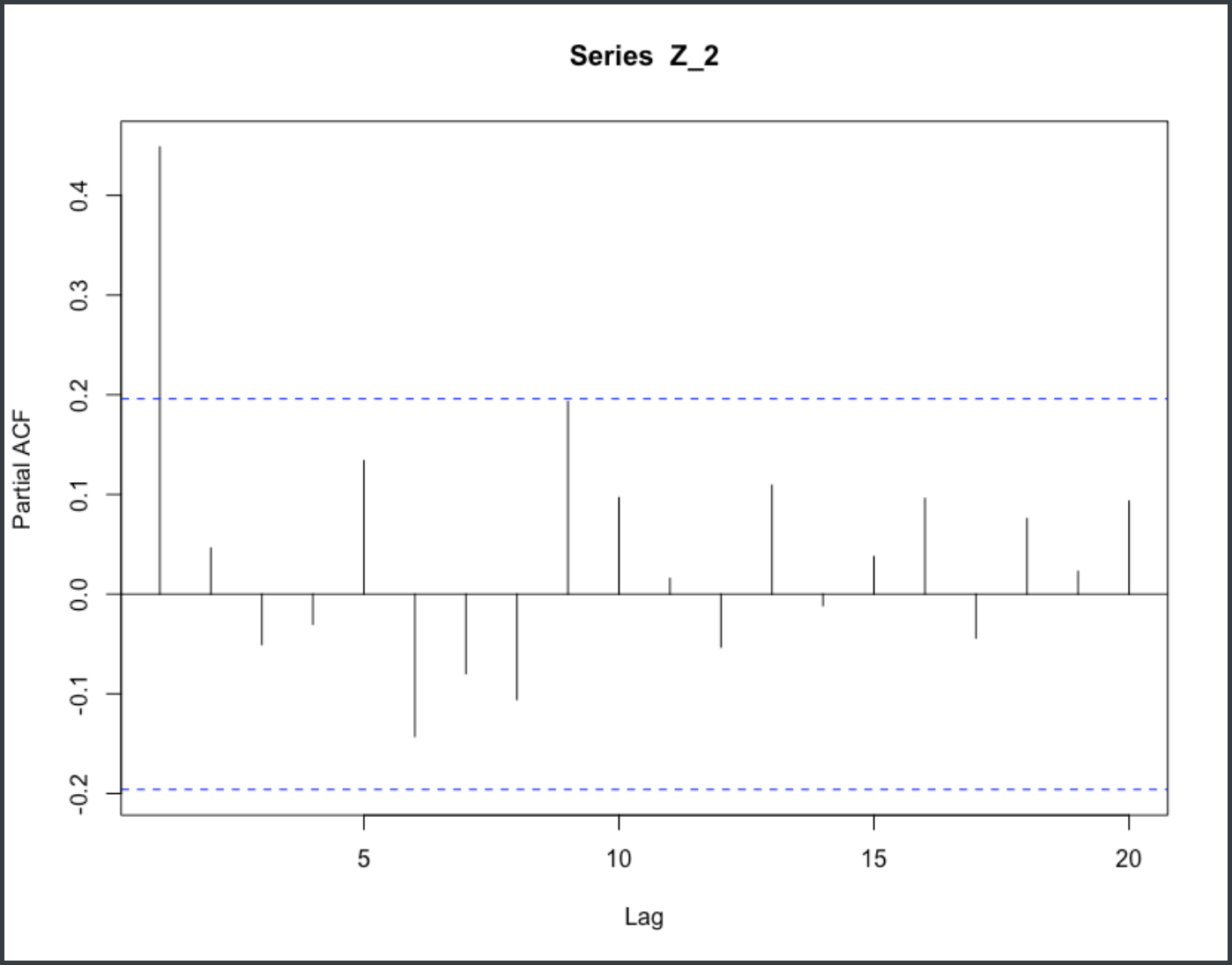
- PACF 함수는 1, 8 시점에서 높은 수치를 보이고 있다.



- 시계열 2
  - 앞선 시계열 1과 마찬가지로 lag 가 3인 시점까지는 자기 상관성이 확인된다.



- PACF 는 안정적이다.



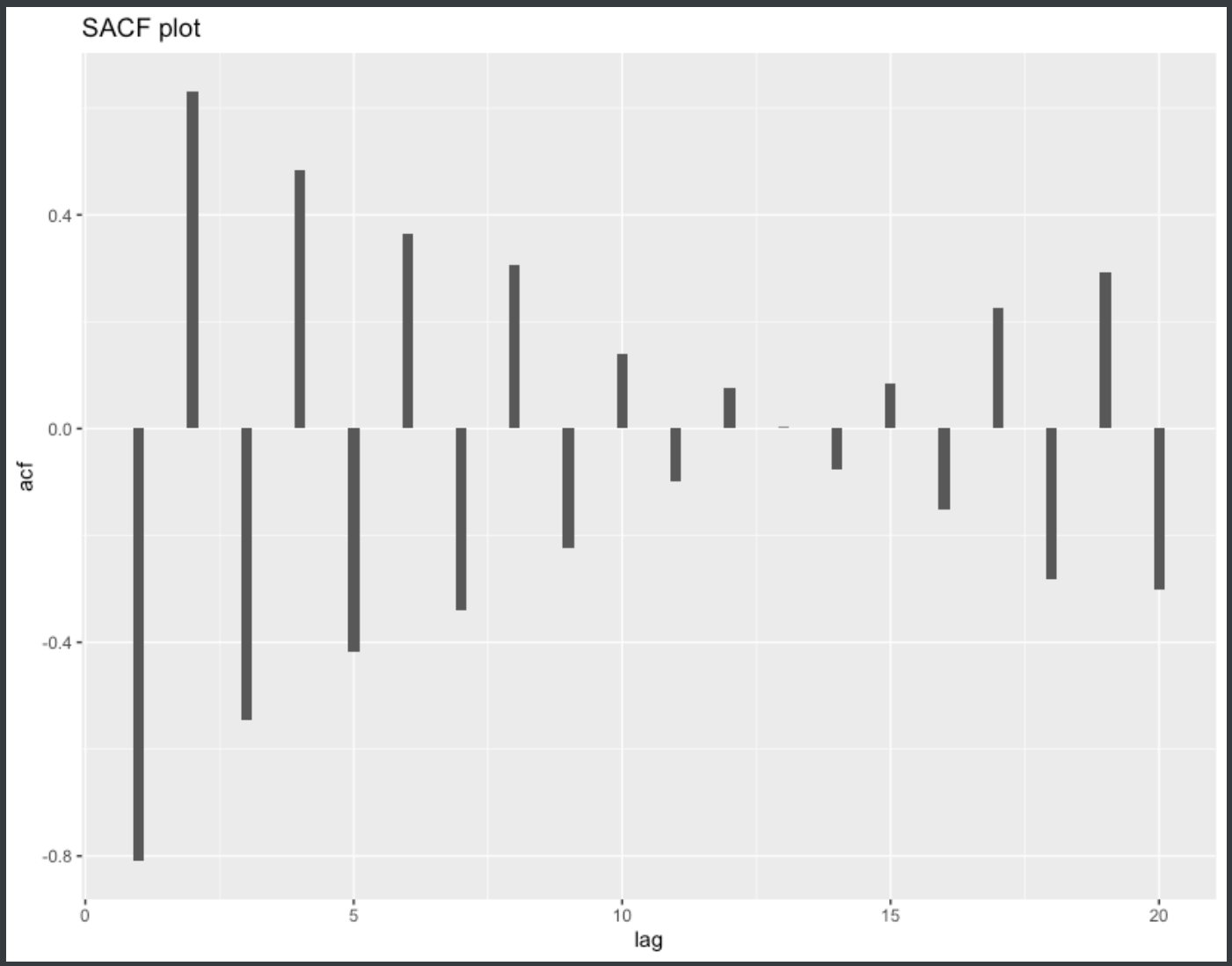
ACF 구현

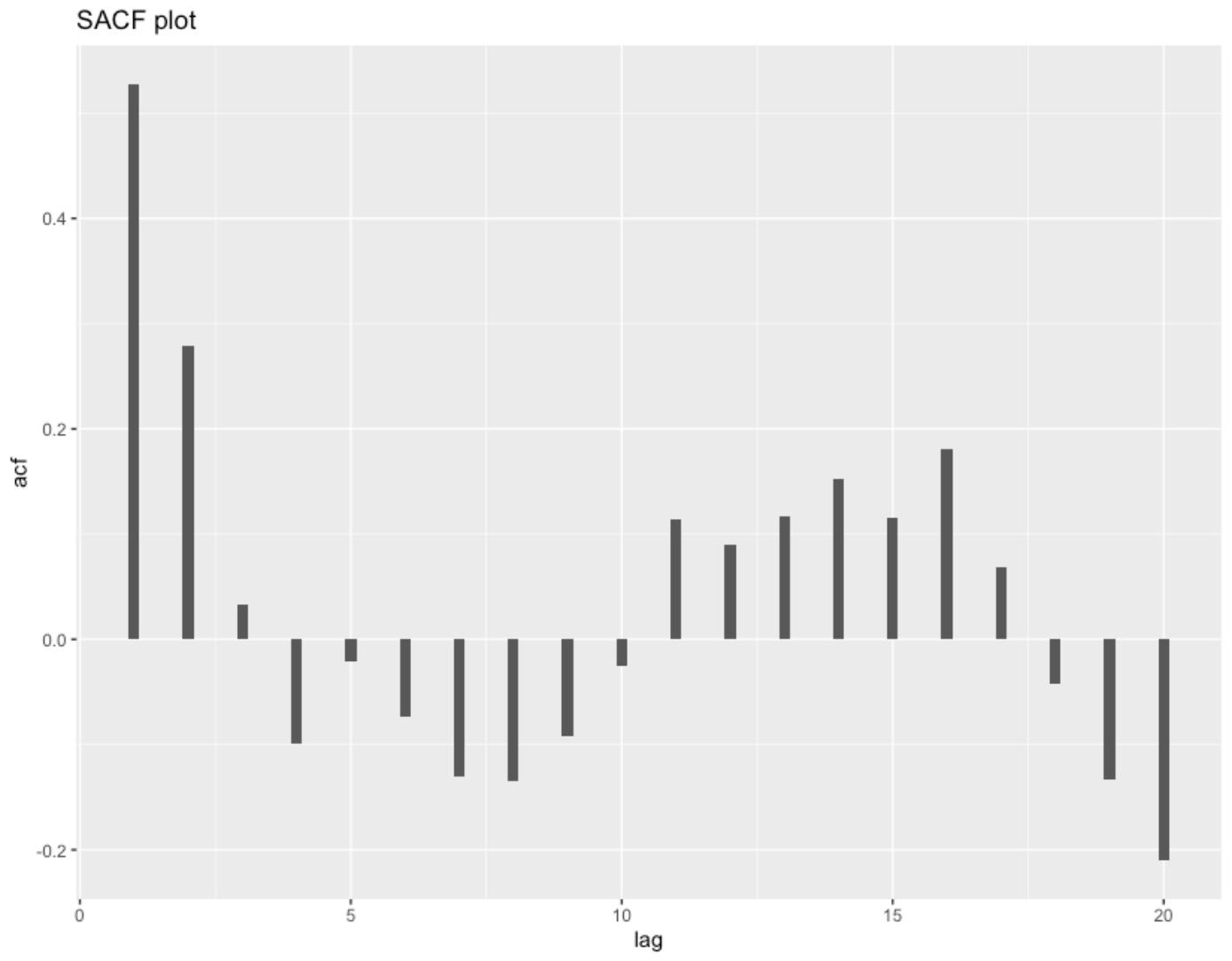
- 먼저 ACF는 다음과 같이 구현하였다.
  - custom_SACF 는 K lag에서의 ACF를 계산하고, draw SACF는 각 k 를 for loop 로 돌면서 20 lag 까지의 결과를 모은다.
  - SACF 함수의 수식은 슬라이드 8을 참고하였다.

```
custom_SACF = function(Z, k) {
  N = length(Z)
  Z_mean = mean(Z)
  Z_var = sum((Z - Z_mean)^2) / N

  term_1 = Z[1:(N-k)] - Z_mean
```

```
  term_2 = Z[(1+k):N] - Z_mean
  u = sum(term_1 * term_2) / N

  return (u / Z_var)
}

draw_SACF = function(Z) {
  results = as.data.frame(
    matrix(nrow=20, ncol=2)
  )
  colnames(results) = c("lag", "acf")

  for (i in 1:20){
    results[i, 1] = i
    results[i, 2] = custom_SACF(Z, i)
  }

  ggplot(data=results, aes(x=lag, y=acf)) +
    geom_bar(stat='identity', width=0.2) +
    ggtitle("SACF plot")
}
```

- 시계열 1번


SACF plot
0.4
0.0
-0.4
-0.8
acf
0
5
10
15
20
lag



PACF 함수 구현

- PACF는 다음과 같이 구현하였다.
  - custom_SACF 는 K lag에서의 PACF를 계산하고, draw SACF는 각 k 를 for loop 로 돌면서 20 lag 까지의 결과를 모은다.
  - SPACF 함수는 durbin-levison 알고리즘으로 구현하기에 어려움이 많아 회귀 모형을 적합하고 잔차의 correlation 을 보는 방식을 채택하였다.
  - 함수 구현을 위해 https://rpubs.com/skkong/pacf_1 자료를 참고하였다.

```
lagmat = function(Z, k){
  lagmat = matrix(nrow=length(Z), ncol=k+1)
  lagmat[1:length(Z), 1] = Z
```

```
  for (i in 1:k){
    Z = lag(Z)
    lagmat[1:length(Z), i+1] = Z
  }

  return (lagmat)
}

custom_SPACF = function(Z, k) {
  lag_mat = lagmat(Z, k)
  X = lag_mat[(k+1):length(Z), 2:k]
  y_forward = lag_mat[(k+1):length(Z), 1]
  y_backward = lag_mat[(k+1):length(Z), (k+1)]
  model_forward = lm(y_forward ~ X)
  model_backward = lm(y_backward ~ X)
  a = as.numeric(model_forward$residuals)
  b = as.numeric(model_backward$residuals)

  return (cor(a, b))
}

draw_SPACF = function(Z) {
  results = as.data.frame(
    matrix(nrow=20, ncol=2)
  )
  colnames(results) = c("lag", "pacf")

  for (i in 1:20){
    results[i, 1] = i
    results[i, 2] = custom_SPACF(Z, i)
  }

  ggplot(data=results, aes(x=lag, y=pacf)) +
    geom_bar(stat='identity', width=0.2) +
    ggtitle("SPACF plot")
}
```

- lagmat 함수는 lag 함수를 이용해서 lag 를 k 만큼 생성하는 작업을 진행한다.
  - 이때 X는 다음과 같이 생성된다.

```
> lagmat(Z_1, 5)
              [,1]        [,2]        [,3]        [,4]
[,5]        [,6]
  [1,]  0.68506053          NA          NA          NA
 NA          NA
  [2,] -0.34779051  0.68506053          NA          NA
 NA          NA
  [3,] -0.03679638 -0.34779051  0.68506053          NA
 NA          NA
  [4,]  0.04149140 -0.03679638 -0.34779051  0.68506053
 NA          NA
  [5,] -1.59063610  0.04149140 -0.03679638 -0.34779051
 0.68506053          NA
  [6,]  1.17043799 -1.59063610  0.04149140 -0.03679638
-0.34779051  0.68506053
  [7,]  0.93544073  1.17043799 -1.59063610  0.04149140
-0.03679638 -0.34779051
  [8,] -2.20903902  0.93544073  1.17043799 -1.59063610
 0.04149140 -0.03679638
```

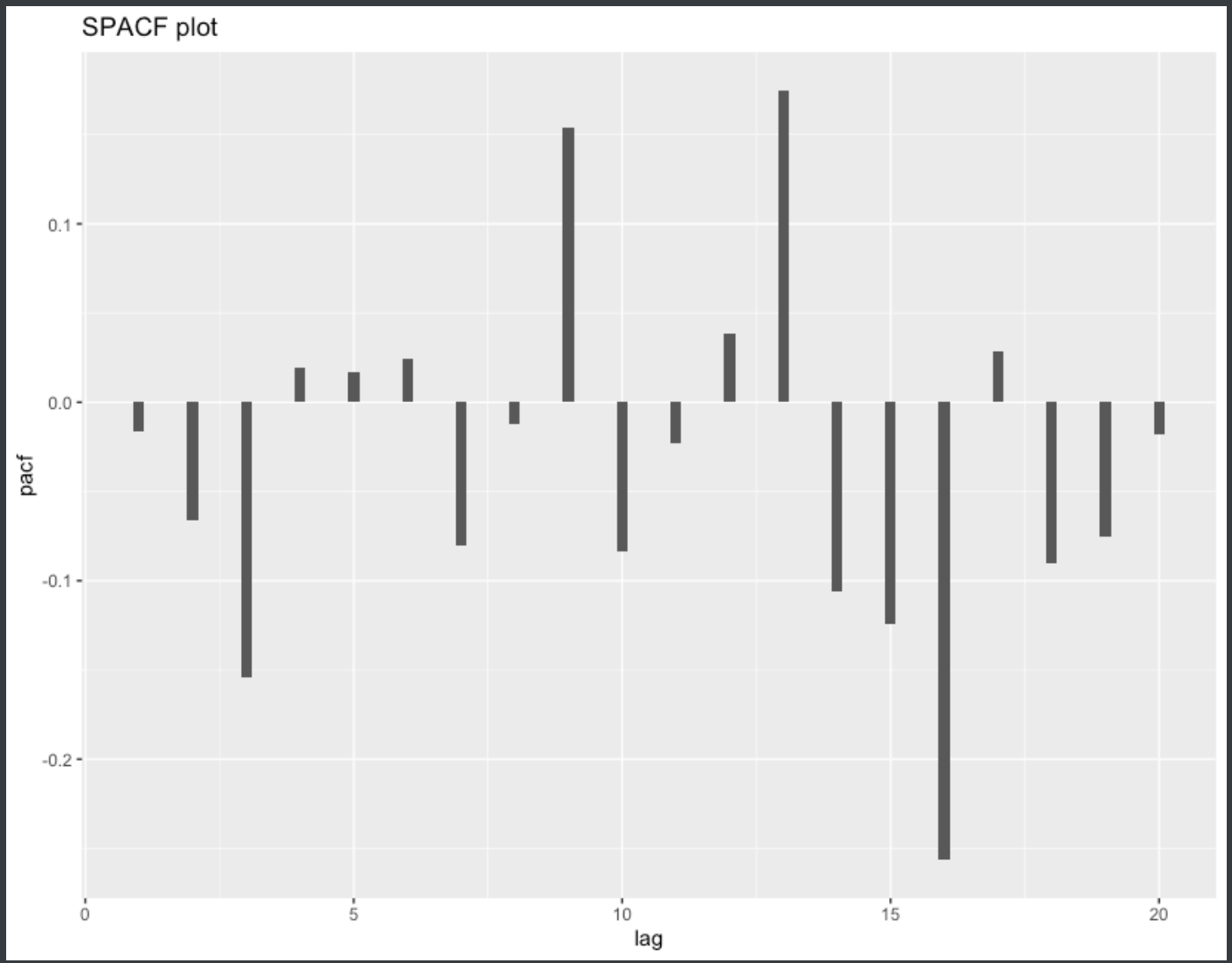
- t 시점과 t+k 시점의 값을 예측하는 모델 2개를 적합하였다. 이때 각 target varable을 예측하기 위해 t+1, t+2, ..., t+k-1의 데이터가 이용된다.

```
X = lag_mat[(k+1):length(Z), 2:k]
y_forward = lag_mat[(k+1):length(Z), 1]
y_backward = lag_mat[(k+1):length(Z), (k+1)]
model_forward = lm(y_forward ~ X)
model_backward = lm(y_backward ~ X)
```

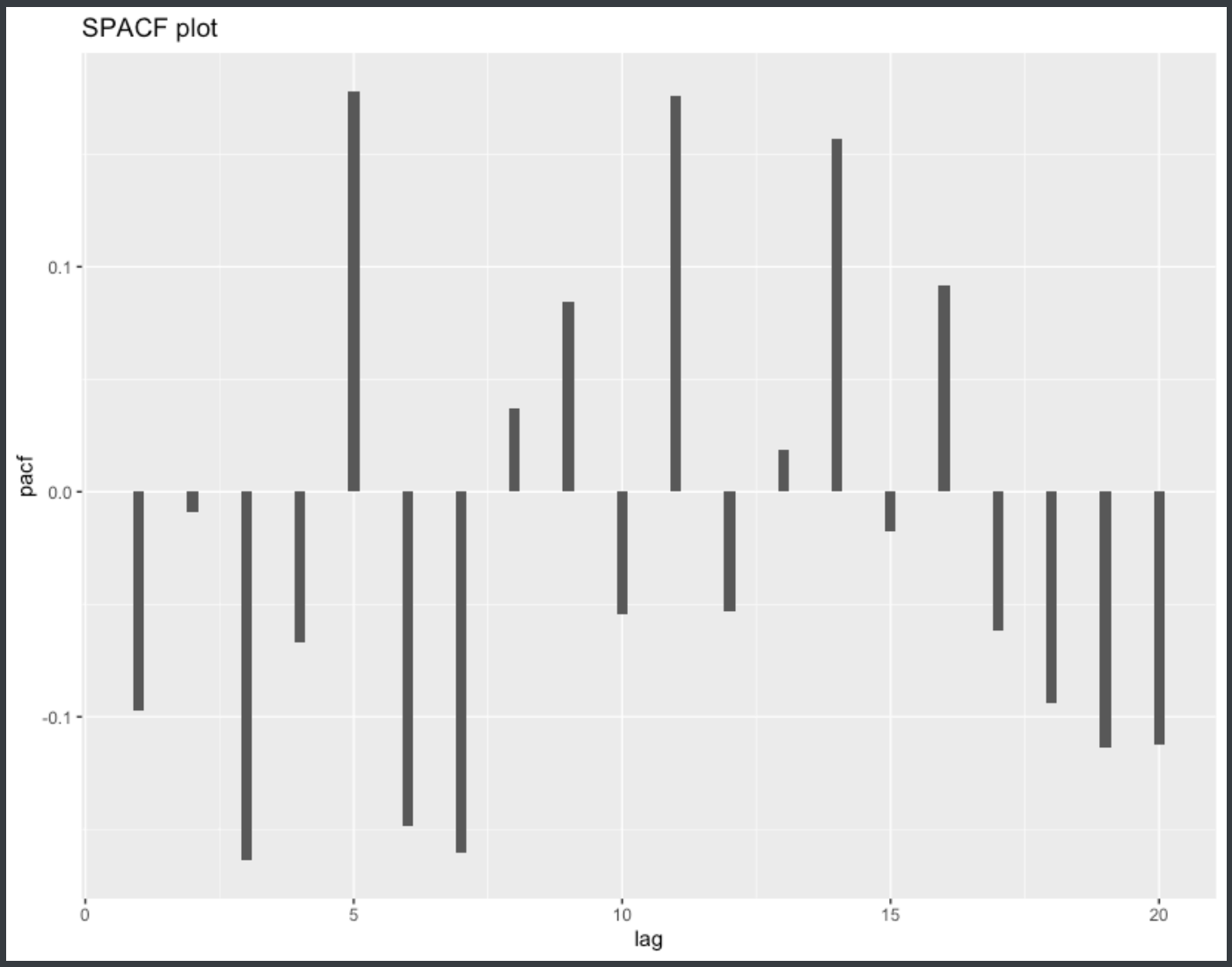
- 두 모형이 적합되고 남은 잔차에 대해 correlation coeficient를 계산하였다.

```
a = as.numeric(model_forward$residuals)
b = as.numeric(model_backward$residuals)

cor(a, b)
```

- 시계열 1번



- 시계열 2번



적합의 결과가 R 내장 함수로 제공되는 PACF와 상이하여 구현 상의 문제가 있는 것으로 추정됩니다. 모델을 적합하기 위해 lag 하는 부분에서 문제가 생긴 것으로 추정하고 있습니다.