R입문 기말고사

이상민(202115813)

12/21/2021

1번 회귀분석

1)

```
e = rnorm(1000, 0, 1)
```

평균이 0, 표준편차가 1인 정규분포로부터 100개의 난수발생=>벡터에 저장

2)

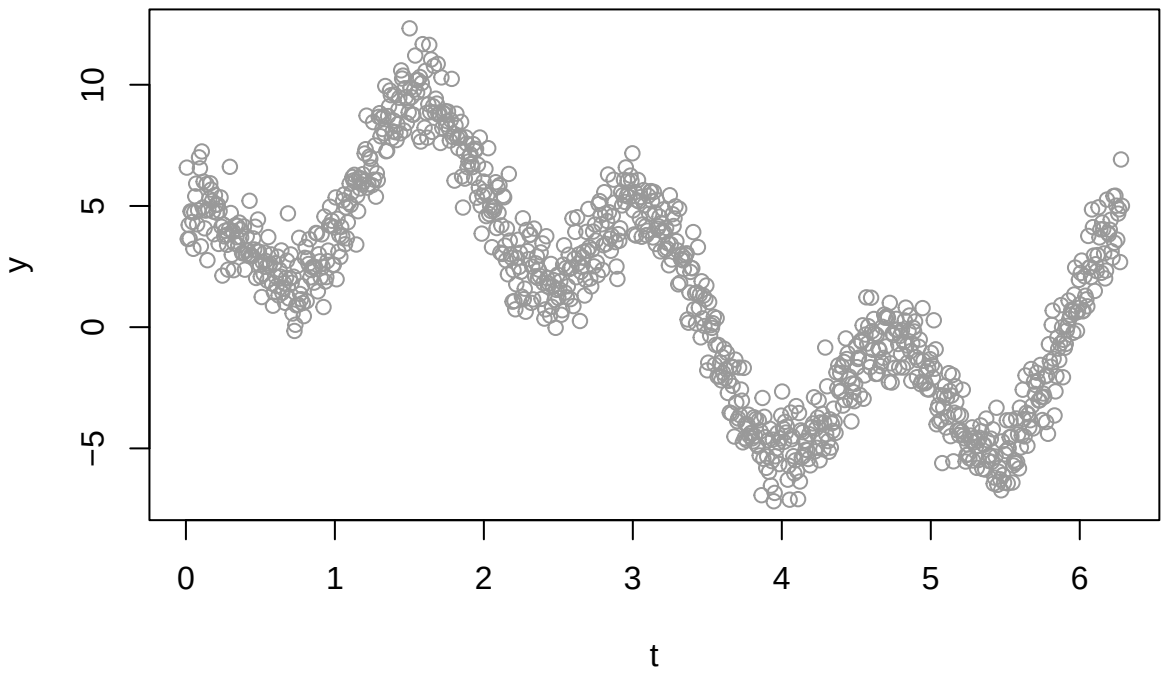
```
i = 1: 1000
t = 2*pi*i/1000
```

t벡터 생성

```
x1 = sin(t)
x2 = cos(4*t)
```

3)

```
y = 1.5 + 5*x1 + 3*x2 + e
plot(x=t, y=y, col='gray60')
```



4)

```
x = cbind(1, x1, x2)
```

5)

```
b = matrix(c(1.5, 5, 3))
```

b라는 매트릭스 만들기

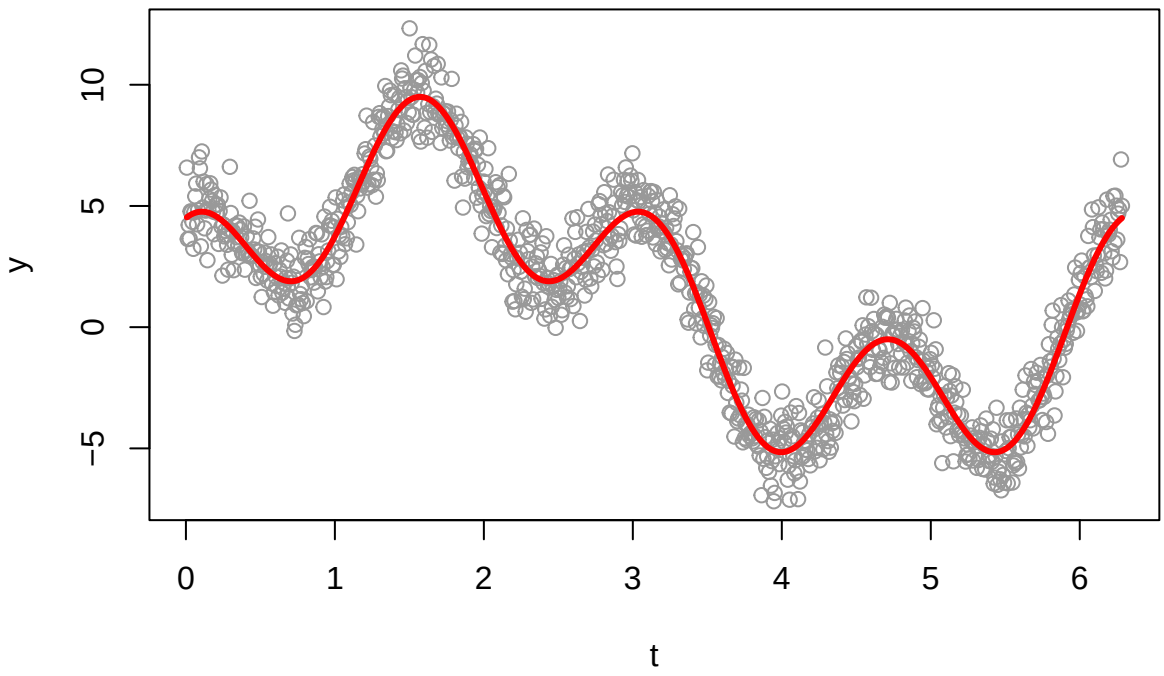
```
xb = x%*%b
```

매트릭스 곱 계산시 %*% (1000x3행렬과 3x1행렬의 곱 => 1000x1행렬)

```
xb = as.vector(xb)
```

계산결과를 벡터화

```
plot(x=t, y=y, col='gray60')
lines(t, xb, col='red', lwd=3)
```



lines함수를 이용하여 산점도에 붉은선 추가. lwd는 선의굵기

6)

```
x_t = t(x)
```

t()는 전치행렬 구하기

```
y = matrix(y)
```

기존 y가 벡터이므로 1000x1행렬로 변환해줌

```
b_hat = solve(x_t%*%x)%*%x_t%*%y
```

solve()는 역행렬 구하기

```
b_hat ; b
```

```
##         [,1]
##    1.499954
## x1 5.050546
## x2 3.099002
```

```
##      [,1]
## [1,]  1.5
```

```
## [2,]  5.0
## [3,]  3.0
```

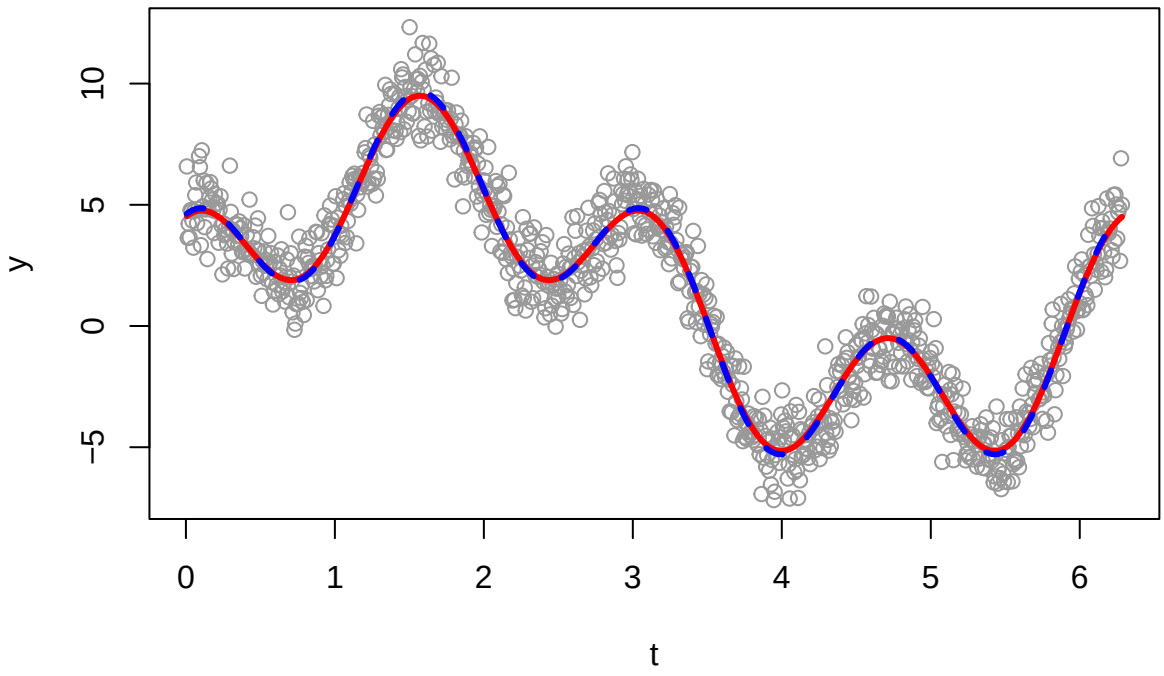
b_hat과 b 비교: b_hat을 소수첫째자리까지 반올림하면 b이다.

7)

```
xb_hat = x%*%b_hat
xb_hat = as.vector(xb_hat)
```

계산결과를 벡터화

```
plot(x=t, y=y, col='gray60')
lines(t, xb, col='red', lwd=3)
lines(t, xb_hat, col='blue', lwd=3, lty=2)
```

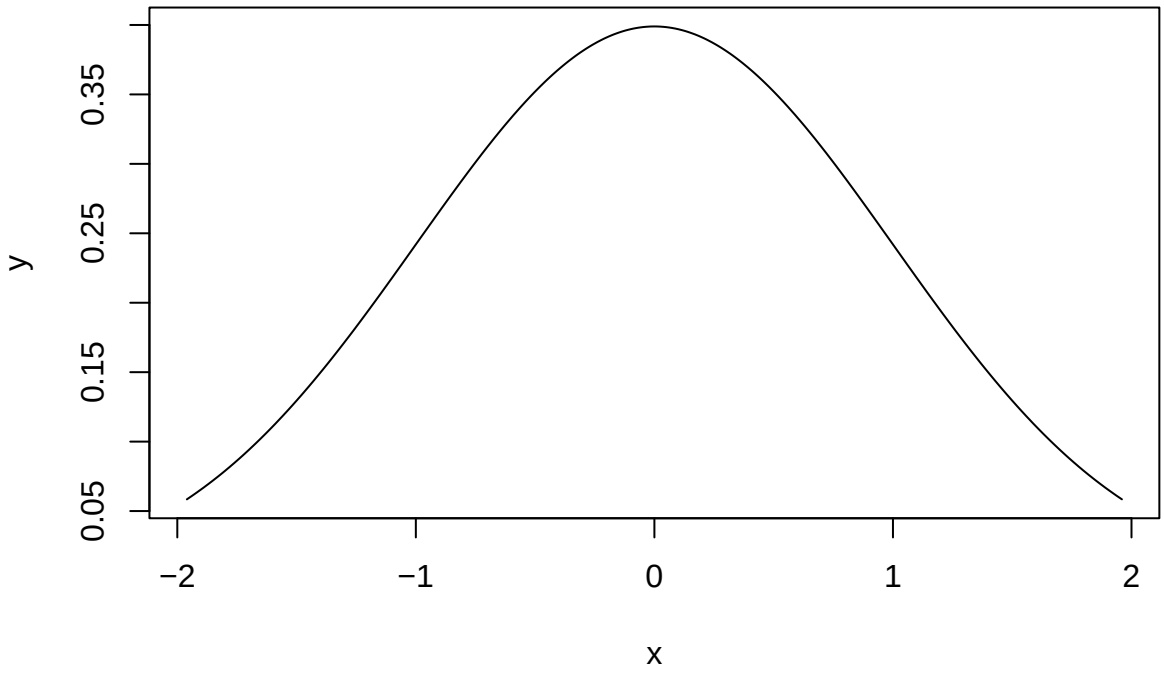


lines함수를 이용하여 산점도에 붉은선 추가. lwd는 선의굵기. lty는 선의모양

2번 몬테카를로 적분

1)

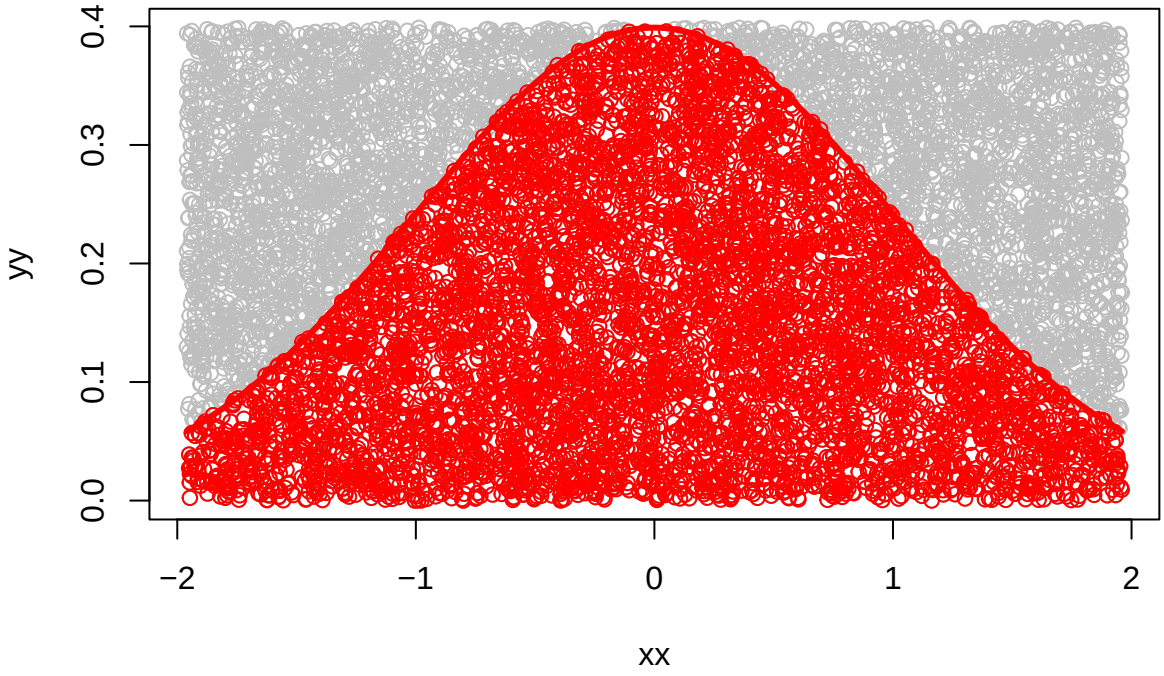
```
x = seq(from=-1.96, to=1.96, by=0.01)
y = exp(-(x^2)/2) / sqrt(2*pi)
plot(x, y, type='l')
```




```
xx = runif(n=10000, min=-1.96, max=1.96)
yy = runif(n=10000, min=0, max=1/sqrt(2*pi))

plot(xx, yy, col='gray')
lines(x,y,col='red', lwd=3)

test=function(xx,yy){
  yy < exp(-(xx^2)/2)/sqrt(2*pi)
}

tst=c()
for (i in 1:10000) tst[i] = test(xx[i], yy[i])
head(tst)
```

```
## [1] FALSE  TRUE  TRUE  TRUE FALSE  TRUE
```

```
points(xx[tst],yy[tst],col='red')
```




```
sum(tst)
```

```
## [1] 5939
```

```
(sum(tst)/10000)*(1.96*2 * (1/sqrt(2*pi)))
```

```
## [1] 0.9287727
```

약 0.95가 나온다.

2)

```
rand = rnorm(1000)
sum(rand>=-1.96 & rand<=1.96)
```

```
## [1] 952
```

약 950번이 나온다.

3번 징검다리

8번 참가자가 살아남을 확률이 더 높은것은 type A, type B 중 평균적으로 8번차례가 되었을 때 남은 유리 개수가 작은것이다.

type A

변수로 ndie(죽은사람의 수), nglass(건넌유리 수), isbreak(유리가 깨졌는지 여부. 0이면 안깨짐, 1이면 깨짐)를 설정하였고, 각 실험마다 처음에 0으로 초기화시켜주었다.

ex) ndie = 0 #죽은사람의 수 nglass = 0 #건넌유리 수 isbreak = 0 #유리가 깨졌는지 여부 (0이면 안깨짐. 1이면 깨짐)

```
a=c()
for (j in 1:10000){
  ndie = 0
  nglass = 0
  isbreak = 0
  for (i in 1:20) {
    if(nglass == 20 || ndie == 2) break;

    if(isbreak == 1) {
      nglass = nglass + 1
      isbreak = 0
      next
    }

    if(ndie != 1) {
      random = rbinom(1, 1, 0.5)
      if(random==0) {
        isbreak=1
        ndie=ndie+1
      }else{
        nglass=nglass+1
      }
    }else{
      random = rbinom(1, 1, 0.95)
      if(random==0) {
        isbreak = 1
        ndie = ndie +1
      }else{
        nglass = nglass +1
      }
    }
```

```
  }
  nglass
  a[j] = nglass
}
sum(a)/10000
```

```
## [1] 13.111
```

1회의 실험(최대 for문 20회 반복)] 처음에 ndie(죽은사람의 수)=0, nglass(건넌유리 수)=0, isbreak(유리가 깨졌는지 여부)를 변수를 초기화 시켜주었다.

이때, 유리의 개수가 총 20개 이므로, for문을 최대 20번 돌게된다.

이 for문은 nglass(건넌유리의 수)가 20이거나 ndie(죽은사람의 수)가 2이면 종료하게 된다.(8번 참가자의 순서가 올때까지!!!)

만약 isbreak = 1이라면(유리가 깨져있다면), 건넌 유리의 수를 1증가 시키고, isbreak =0으로(유리가 안깨짐)으로 바꾸고, 다시 for문의 처음으로 가게 된다.

죽은 사람의 수가 1이 아니라면(10번 참가자라면), rbinom(1, 1, 0.5)에서 random하게 1개를 뽑게된다.

이때 0이(일반유리) 나오면, ndie(죽은사람의 수)를 1로 증가시키고, isbreak = 1로(유리가 깨짐) 변경하게 된다.

이때 1이(강화유리) 나오면, nglass(건넌유리 수)를 1 증가시긴다..

죽은 사람의 수가 1이라면(9번 참가자라면), rbinom(1, 1, 0.95)에서 random하게 1개를 뽑게 된다.

이때 0이(일반유리) 나오면, ndie(죽은사람의 수)를 1로 증가시키고, isbreak = 1로(유리가 깨짐) 변경하게 된다.

이때 1이(강화유리) 나오면, nglass(건넌유리 수)를 1 증가시킨다.

‘_____’ 위와 같은 실험을 10000번을 반복하며, 평균을 구해보았다.

type B

```
a=c()
for (j in 1:10000){
  ndie = 0
  nglass = 0
  isbreak = 0
  for (i in 1:20) {
    if(nglass == 20 || ndie == 7) break;
    if(isbreak == 1) {
      nglass = nglass + 1
```

```
      isbreak = 0
      next
    }

    random = rbinom(1, 1, 0.5)
    if(random==0) {
      isbreak=1
      ndie=ndie+1
    }else{
      nglass=nglass+1
    }
  }
  a[j] = nglass
}
sum(a)/10000
```

```
## [1] 12.1538
```

1회의 실험(최대 for문 20회 반복)] 처음에 ndie(죽은사람의 수)=0, nglass(건넌유리 수)=0, isbreak(유리가 깨졌는지 여부)를 변수를 초기화 시켜주었다.

이때, 유리의 개수가 총 20개 이므로, for문을 최대 20번 돌게된다.

이 for문은 nglass(건넌유리의 수)가 20이거나 ndie(죽은사람의 수)가 7이면 종료하게 된다.(8번 참가자의 순서가 올때까지!!!)

만약 isbreak = 1이라면(유리가 깨져있다면), 건넌 유리의 수를 1증가 시키고, isbreak =0으로(유리가 안깨짐)으로 바꾸고, 다시 for문의 처음으로 가게 된다.

rbinom(1, 1, 0.5)에서 random하게 1개를 뽑는다. 이때 0이(일반유리) 나오면, ndie(죽은사람의 수)를 1로 증가시키고, isbreak = 1로(유리가 깨짐) 변경하게 된다. 이때 1이(강화유리) 나오면, nglass(건넌유리 수)를 1 증가시킨다.

‘————————————————————’ 위와 같은 실험을 10000번을 반복하며, 평균을 구해보았다.

type A에서 8번 참가자의 순서가 올때까지 건넌 다리의 수가 평균 13개 정도이고 type B에서 8번 참가자의 순서가 올때까지 건넌 다리의 수가 평균 12개 정도 나왔다.

따라서 남은 유리의 개수는 type A는 약 20-13=7개 type B는 약 20-12=8개 이므로 type A가 더 적으므로 type A가 8번 참가자가 살아남을 확률이 높다.

답: type A

4번 COVID19

데이터 불러오기

install.packages(“readr”) 만약 패키지 설치 안되어있다면 설치

```
library(readr)
df = read_csv('https://raw.githubusercontent.com/guebin/2021IR/master/_notebooks/covid19
```

```
## Rows: 12294 Columns: 5
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): prov
## dbl (4): year, month, day, cases
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(df)
```

```
## # A tibble: 6 x 5
##    year month   day prov  cases
##   <dbl> <dbl> <dbl> <chr> <dbl>
## 1  2020     1    20 서울      0
## 2  2020     1    20 부산      0
## 3  2020     1    20 대구      0
## 4  2020     1    20 인천      1
## 5  2020     1    20 광주      0
## 6  2020     1    20 대전      0
```

1)

```
df2020 = df[df$year == 2020, ]
```

2020년의 확진자 자료만 뽑음

```
sum(df2020$cases)
```

```
## [1] 60726
```

2020년의 확진자(cases)총합

```
df2021 = df[df$year == 2021, ]
```

2021년의 확진자 자료만 뽑음

```
sum(df2021$cases)
```

```
## [1] 396886
```

2021년의 확진자(cases)총합

2)

```
df01_15 = df2020[df2020$month ==2 & df2020$day>=1 & df2020$day<=15, ]
```

2020년 2월 1일 ~ 2020년 2월 15일까지의 자료만 뽑음

sol 1)

```
tapply(df01_15$cases, df01_15$prov, sum)
```

```
## 강원 검역 경기 경남 경북 광주 대구 대전 부산 서울 세종 울산 인천 전남 전
북 제주
##    0    0    9    0    0    2    0    0    0    5    0    0    0    1    0    0
## 충남 충북
##    0    0
```

tapply함수를 통해 자료를 지역별로(df01_15$prov) 나누어 함수(sum) 적용

sol 2)

```
f_prov = factor(df01_15$prov)
```

지역을 factor로 변환

```
aggregate(df01_15$cases, list(f_prov), sum)
```

```
##    Group.1 x
## 1     강원 0
## 2     검역 0
## 3     경기 9
## 4     경남 0
## 5     경북 0
## 6     광주 2
## 7     대구 0
## 8     대전 0
## 9     부산 0
## 10    서울 5
## 11    세종 0
## 12    울산 0
## 13    인천 0
## 14    전남 1
## 15    전북 0
## 16    제주 0
## 17    충남 0
## 18    충북 0
```

aggregate함수를 통해 데이터 프레임을 지역별로(list(f_prov)) 나누어 함수(sum) 적용

결과 : 2020년 2월 1일 ~ 2020년 2월 15일까지 가장 많은 확진자가 발견된 지역은 경기임을 알 수 있다.

3)

```
df16_29 = df2020[df2020$month ==2 & df2020$day>=15 & df2020$day<=29, ]
```

2020년 2월 16일 ~ 2020년 2월 29일까지의 자료만 뽑음

sol 1)

```
tapply(df16_29$cases, df16_29$prov, sum)
```

```
## 강원 검역 경기 경남 경북 광주 대구 대전 부산 서울 세종 울산 인천 전남 전
북 제주
##    7    0   65   59  472    7 2055   13   75   62    1   17    5    1    4    2
## 충남 충북
##   48   10
```

tapply함수를 통해 자료를 지역별로(df16_29$prov) 나누어 함수(sum) 적용

sol 2)

```
f_prov = factor(df16_29$prov)
```

지역을 factor로 변환

```
aggregate(df16_29$cases, list(f_prov), sum)
```

```
##    Group.1    x
## 1     강원    7
## 2     검역    0
## 3     경기   65
## 4     경남   59
## 5     경북  472
## 6     광주    7
## 7     대구 2055
## 8     대전   13
## 9     부산   75
## 10    서울   62
## 11    세종    1
## 12    울산   17
## 13    인천    5
## 14    전남    1
## 15    전북    4
## 16    제주    2
```

```
## 17     충남    48
## 18     충북    10
```

aggregate함수를 통해 데이터 프레임을 지역별로(list(f_prov)) 나누어 함수(sum) 적용

결과 : 2020년 2월 16일 ~ 2020년 2월 29일까지 가장 많은 확진자가 발견된 지역은 대구임을 알 수 있다.