통계계산 과제 1

통계학과 2018580018 윤주연

1. 0.2와 0.3-0.1에 대하여 컴퓨터 연산에서도 등식이 정확히 성립하는가?
(R에서 == 연산자와 all.equal 함수 이용하기)
또한 실수 값을 비교할 때 어떻게 해야 하는지 설명하시오.

< R 입력 값 & 결과 값>
> 0.2 == (0.3-0.1)
[1] FALSE
> all.equal(0.2,(0.3-0.1))
[1] TRUE

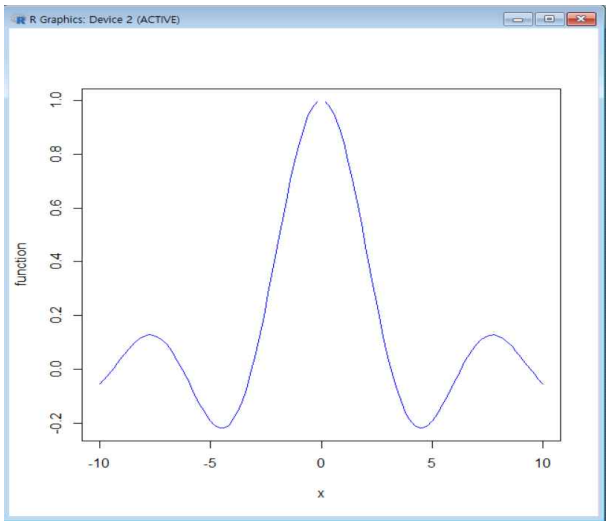
-> == 연산자는 정확하게 이 두 숫자가 같은지를 비교한다. all.equal 함수는 실수의 비교, 즉, 오차 범위 내에서 두 실수가 같은지를 비교한다. floating point 연산에서는 표현 방식에 따라 오차가 생길 수 있다. 따라서 == 연산자로 등식이 정확히 성립하는지를 알 수 없다.

2. 함수 $f(x) = \frac{\sin x}{x}$ 를 n개의 점을 이용하여 그려주는 프로그램 evaluatefunctiosinc를 작성하시오. ($x \in [x_{\min}, x_{\max}]$)
evaluatefunctionsinc(-10,10,100)
evaluatefunctionsinc(-10^-20, 10^-20, 100)
실행하고, 불연속점 x = 0 관찰하시오.
좁은 범위의 $x \in [-\epsilon, \epsilon]$에 대하여 $\lim_{x \to 0} f(x) = 1$ 이 되도록 evaluatefunctionsincwithcheck 작성하시오.
evauatefunctionsincwithcheck(-10^-20, 10^20, 100, 10^-30)으로 x = 0에서 수치적인 문제 해결되었는지 살펴보시오.

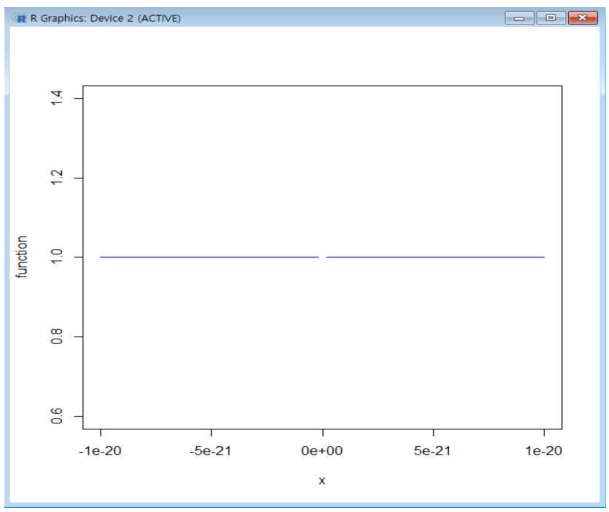
① 프로그램 evaluatefuntionsinc 함수 생성.

```
evaluatefunctionsinc <- function(xmin, xmax, n){
  x = c(0)
  f = c(0)
  for(i in 0:n){
    x[i+1] = xmin + i*(xmax-xmin)/n
    f[i+1] = sin(x[i+1])/x[i+1]
  }
  plot(x,f,type = "l", col= "blue", xlab = "x", ylab = "function")
}
```

② > evaluatefunctionsinc(-10,10,100)



> evaluatefunctionsinc(-10^(-20), 10^(-20), 100)

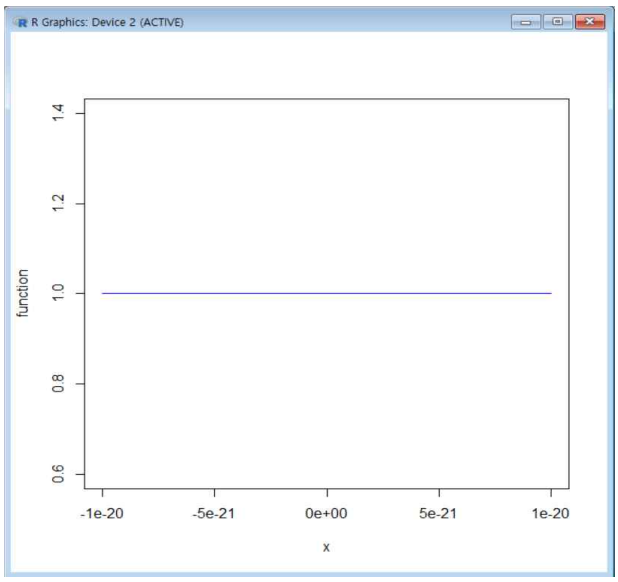


-> x = 0에서 불연속이고 함숫값이 존재하지 않는다.

③ 프로그램 evaluatefunctionsincwithcheck 함수 생성.

```
evaluatefunctionsincwithcheck <- function(xmin, xmax, n, epsilon){
  x = c(0)
  f = c(0)
  for(i in (0:n)){
    x[i+1] = xmin + i*(xmax-xmin)/n
    if(abs(x[i+1]) > epsilon){
      f[i+1] = sin(x[i+1])/x[i+1]
    }
    else{
      f[i+1] = 1
    }
    plot(x, f, type = "l", col = "blue", xlab = "x", ylab = "function")
  }
}
```

④ > evaluatefunctionsincwithcheck(-10^(-20), 10^(-20),100,10^(-30))



-> x = 0값에서 1의 값을 가지므로 수치적인 문제가 해결되었다.

3. 피보나치수열 계산하는 재귀 프로그램과 반복 프로그램을 이용하여, i = 10, 20, 30, 40에 대하여 두 프로그램의 실행속도를 비교하시오. (system.time 함수 이용하기)

< R 입력 값 & 결과 값>

① 재귀 프로그램

```
fiborecursive = function(i){
  if(i <= 2){
    value = 1
  }
  else{
   return(fiborecursive(i-1)+fiborecursive(i-2))
  }
}
```

② 반복 프로그램

```
fiboiterative = function(i){
  if(i <= 2){
    value = 1
  }
  else{
    value1 = 1
    value2 = 1
    for(j in 3:i){
      value <- value1 + value2
      value1 <- value2
      value2 <- value
    }
  }
  return(value)
}
```

③ 실행속도 비교

i = 10

```
> system.time(fiborecursive(10))
```

```
사용자   시스템 elapsed
       0       0       0
> system.time(fiboiterative(10))
 사용자   시스템 elapsed
       0       0       0
```

```
i = 20
> system.time(fiborecursive(20))
 사용자   시스템 elapsed
   0.02    0.00    0.01
> system.time(fiboiterative(20))
 사용자   시스템 elapsed
       0       0       0
```

```
i = 30
> system.time(fiborecursive(30))
 사용자   시스템 elapsed
    0.8     0.0     0.8
> system.time(fiboiterative(30))
 사용자   시스템 elapsed
       0       0       0
```

```
i = 40
> system.time(fiborecursive(40))
 사용자   시스템 elapsed
  94.47    0.47   94.98
> system.time(fiboiterative(40))
 사용자   시스템 elapsed
       0       0       0
```

-> i가 커질수록 재귀 프로그램이 반복 프로그램보다 실행되는 시간이 더 길다. 따라서 반복 프로그램이 실행속도가 재귀 프로그램보다 더 빠르다는 것을 알 수 있다.