PR7 - 함수

조성우

2020년4월30일

함수와 사용자정의 함수

함수

- 특정 목적에 맞게 생성된 연산과정의 집합
- ex) mean 함수: 모든 원소의 합을 원소의 개수로 나눔

##사용자 정의 함수* 사용자의 편의에 따라 직접 작성하여 사용하는 함수* 함수명<- function(인수){ 연산과정} 형태로 작성(한가지만 연산할 경우{} 로 묶지 않아도 됨) * 연산과정으로 나오는 결과값을return,print,cat등으로 반환하는 형태가 이상적

예시1. 두 숫자를 비교해 더큰수를 반환하는 함수

```
#2 개의 숫자를 인수로 받아서 더 큰수를 반환하는 함수
compare <- function(x,y) if(x>y) cat(x) else cat(y)
compare(10,20)
```

```
## 20
```

예시2. 평균값과 표준오차를 계산하는 함수

```
# 표준오차 =  표준편차 /  표본의크기

se <- function(x){
  tmp.sd <-sd(x) # 표준편차
  tmp.N <- length(x) # 표본크기
  tmp.se <- tmp.sd / sqrt(tmp.N) # 평균의 표준오차
}

A <- c(1,2,3,4,5,6,7,8,9,10)
cat(se(A))
```

```
## 0.9574271
```

예시3. 데이터 프레임의 앞뒤 3개의 데이터를 리스트로 보여주는 함수

```
head_tail <- function(x){
  front <- head(x,3)
  rear <- tail(x,3)
  F_R <- list(front,rear) #2개의 데이터프레임 릿트로 묶음
  print(F_R) #묶은 리스트 반환
}

head_tail(mtcars)
```

```
## [[1]]
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
##
## [[2]]
##                mpg cyl disp  hp drat   wt qsec vs am gear carb
## Ferrari Dino  19.7   6  145 175 3.62 2.77 15.5  0  1    5    6
## Maserati Bora 15.0   8  301 335 3.54 3.57 14.6  0  1    5    8
## Volvo 142E    21.4   4  121 109 4.11 2.78 18.6  1  1    4    2
```

예시4. 홀수 판별 함수

```
oddnum <- function(x){
  if(x%%2==1){ #2로 나눈 나머지가 1이면
    return(T)
  }else { #그렇지 않으면
    return(f)} #F를 반환
}
```

scope of variable

- 함수 바깥에서 생성된 변수는 같은함수 안에서는 언제나 사용가능
- 함수 안에서 생성된 변수는 함수가 종료되면 사라짐(local variable은 휘발성)
- 함수 내에서 생성된 변수가 사라지지 않게 하려면 "<<-"을 할당 연산자로 사용(global variable로 할당)

```
# <- 할당 연산자 사용
scopetest <-function(x){
  a <- 10
  print(a)
  print(x)
}

scopetest(9)
```

```
## [1] 10
## [1] 9
```

```
#print(a) #주석 제거후 함수실행하여 메세지 확인할것

# <<- 할당 연산자 사용
scopetest <- function(x){
  a <<- 10
  print(a)
  print(x)
}

scopetest(9)
```

```
## [1] 10
## [1] 9
```

```
print(a)
```

```
## [1] 10
```

함수의default 값 설정

- 인수를 입력하지 않았을때 자동으로 적용되는 값을default라고 함
- 함수작성시 "인수=T 또는 인수=10" 이런식으로 미리 인수에 적용될 값을 입력

```
add10 <- function(x=10)x+10
add10()
```

```
## [1] 20
```

PR7 연습문제

문제1

- PR3의 연습문제1번을 활용한 문제입니다.
- 벡터prices에 저장된 값은2020-03-01 부터2020-03-06 까지bitcoin의 종가이다. *prices <- c(11905000.0, 1973000.0, 12190000.0, 12700000.0, 12303000.0, 12604000.0)
- 힌트를 참고하여 순서대로2020-03-02 부터2020-03-06 까지5 일간의 수익률을 구하는in_rate 함수를 작성하세요.
- 함수를 사용하여 증가율을 출력해 주세요. *HINT 수익율= ((금일의 종가- 전일의 종가)/ 전일의 종가)*100

```
prices <- c(11905000.0, 11973000.0, 12190000.0, 12700000.0, 12303000.0,
12604000.0)
```

```r
in.rate <- function(a){              #Define
  price_today <- a[-1]
  price_yesterday <- a[-length(a)]
  returns <- (price_today - price_yesterday) / price_yesterday * 100
  return(returns)
}

in.rate(prices)
```

```
## [1]  0.5711886  1.8124113  4.1837572 -3.1259843  2.4465578
```

문제2

- PR5의 연습문제 1번을 활용하는 문제입니다.
- URL을 입력받아서 해당 웹사이트의 Table list를 반환하는 함수read.html.tables를 작성하세요.
- 웹사이트의 URL을 입력하여 결과값을 저장하고, 특정 테이블을 head를 사용해서 출력해주세요.(웹사이트 자율)

```r
  library(XML)
```

```
## Warning: package 'XML' was built under R version 3.6.3
```

```r
  library(httr)
```

```
## Warning: package 'httr' was built under R version 3.6.3
```

```r
read.html.tables <- function(URL){      #웹의 content를 불러오는 read.html.tables 함수를 Define 해줍니다.

  html_source <- GET(URL)
  tabs <- readHTMLTable(rawToChar(html_source$content), stringsAsFactors = F)
  return(tabs)
}

KoreaPopulation <-read.html.tables("https://www.worldometers.info/world-population/south-korea-population/") #read.html.tables 함수를 call 하여 변수에 할당
ForecastKorPOP <- KoreaPopulation[2]       #테이블 리스트중 한국 인구예측 테이블을 서브세팅해줍니다.

names(ForecastKorPOP) <- ForecastKorPOP     #해당 테이블의 이름이 NULL 이므로 이름을 지어줍니다.

ForecastKorPOP
```

```
## $...
##    Year Population Yearly %  Change Yearly Change Migrants (net) Median
Age
## 1  2020 51,269,185           0.09 %        43,877         11,731
43.7
## 2  2019 51,225,308           0.10 %        53,602         11,731
41.4
## 3  2018 51,171,706           0.15 %        75,291         11,731
41.4
## 4  2017 51,096,415           0.22 %       112,958         11,731
41.4
## 5  2016 50,983,457           0.32 %       160,364         11,731
41.4
## 6  2015 50,823,093           0.51 %       255,491         80,237
40.8
## 7  2010 49,545,636           0.34 %       168,913        -31,309
38.0
## 8  2005 48,701,073           0.55 %       264,366         16,245
34.8
## 9  2000 47,379,241           0.90 %       417,344         31,886
31.9
## 10 1995 45,292,522           1.08 %       474,821         14,284
29.3
## 11 1990 42,918,419           1.02 %       422,803         34,116
27.0
## 12 1985 40,804,402           1.41 %       551,759         18,578
24.3
## 13 1980 38,045,607           1.46 %       533,389        -33,027
22.1
## 14 1975 35,378,661           1.90 %       636,596        -41,988
19.9
## 15 1970 32,195,681           2.19 %       660,025        -16,369
19.0
## 16 1965 28,895,558           2.67 %       713,209        -13,827
18.4
## 17 1960 25,329,515           3.32 %       762,989         62,079
18.6
## 18 1955 21,514,570           2.29 %       460,637         86,590
18.9
##    Fertility Rate Density (P/Km짼) Urban Pop % Urban Population
## 1            1.11              527      81.8 %       41,934,110
## 2            1.21              527      81.6 %       41,805,375
## 3            1.21              526      81.4 %       41,678,226
## 4            1.21              526      81.3 %       41,552,264
## 5            1.21              524      81.3 %       41,426,777
## 6            1.23              523      81.3 %       41,301,851
## 7            1.17              510      81.9 %       40,601,614
## 8            1.21              501      81.4 %       39,622,010
## 9            1.50              487      79.6 %       37,729,427
```

```
## 10           1.68               466      78.2 %       35,441,319
## 11           1.57               441      73.9 %       31,696,103
## 12           2.23               420      64.9 %       26,474,831
## 13           2.92               391      56.7 %       21,582,191
## 14           4.00               364      48.0 %       16,997,155
## 15           4.65               331      40.7 %       13,110,502
## 16           5.60               297      32.4 %        9,351,713
## 17           6.33               261      27.7 %        7,022,058
## 18           5.65               221      24.4 %        5,251,885
##    Country's Share of World Pop World Population South KoreaGlobal Rank
## 1                         0.66 %    7,794,798,739                    28
## 2                         0.66 %    7,713,468,100                    28
## 3                         0.67 %    7,631,091,040                    28
## 4                         0.68 %    7,547,858,925                    27
## 5                         0.68 %    7,464,022,049                    27
## 6                         0.69 %    7,379,797,139                    27
## 7                         0.71 %    6,956,823,603                    26
## 8                         0.74 %    6,541,907,027                    25
## 9                         0.77 %    6,143,493,823                    24
## 10                        0.79 %    5,744,212,979                    24
## 11                        0.81 %    5,327,231,061                    24
## 12                        0.84 %    4,870,921,740                    23
## 13                        0.85 %    4,458,003,514                    23
## 14                        0.87 %    4,079,480,606                    23
## 15                        0.87 %    3,700,437,046                    24
## 16                        0.87 %    3,339,583,597                    24
## 17                        0.83 %    3,034,949,748                    24
## 18                        0.78 %    2,773,019,936                    24
```

문제3

- 어떤 문제를 수치 계산으로 풀지 않고 확률(난수)를 이용해서 푸는 것을 몬테카를로법이라 한다.이방법으로 원주율 파이를 구할 수 있다. **각 도형의 면적** * 반지름이 1 인 원 면적의 1/4 = 1/4 x pi X 1^2 = pi/4 * 한 변이 1 인 정사각형의 면적 = 1^2 = 1 **공식 유도** 1. 1/4 원 내부에 표시된 난수 개수 : a 2. 1/4 원 외부에 표시된 난수 개수 : b 3. pi/4:1 = a:a+b 4. pi = 4a/a+b = 4a/n **이미지는 PR 본문참고**

  - 점을 뽑을 갯수 n 을 입력받아 파이를 추정하는 함수 mc_pi 를 작성하세요.
  - 점을 100 개,1000 개,10000 개,100000 개 뽑았을 때 추정된 파이를 출력하고 실제 파이값과 가까워지는지 확인하세요. * HINT1 : 난수생성함수 runif(n,min=0,max=1) * 0 에서 1 사이의 난수를 n 개 생성 (default)-+ HINT2 : 좌표평면상의 점이므로, 각각 n 개의 x 값과 y 값이 필요합니다. 두점과 원점 (0,0) 사이의 거리는 다음과 같이 구할 수 있습니다.

```
    *distance = sqrt(x^2 + y^2)+
```

```r
vec <- c(100,1000,10000,100000,1000000)

mc_pi <- function(n) {

  x <- runif(n, min=0, max=1) #좌표평면상 x좌표가 0~1 인 x  난수생성
  y <- runif(n, min=0, max=1) #좌표평면상 y좌표가 0~1 인 y  난수생성

  #밑의 두줄코드는 난수와 원점사이의 거리공식,pi  공식을 통해 pi를 구하는 과정입니다
  xy <- ifelse(x^2 + y^2 <= 1,T,F)  #x제곱과 y제곱의 합이 1보다 작거나 같은 경우 xy에
count
  PI <- 4*(sum(xy)/n)   #pi값을 구하는 공식 적용
  return(PI)
}


cat("n=",100,"일 때 추정된 PI : ",mc_pi(vec[1]),"\n","n=",1000,"일 때 추정된 PI :
",mc_pi(vec[2]),"\n","n=",10000,"일 때 추정된 PI :
",mc_pi(vec[3]),"\n","n=",100000,"일 때 추정된 PI :
",mc_pi(vec[4]),"\n","n=",1000000,"일 때 추정된 PI : ",mc_pi(vec[5]),"\n")
```

```
## n= 100 일 때 추정된 PI :  3.12
##  n= 1000 일 때 추정된 PI :  3.128
##  n= 10000 일 때 추정된 PI :  3.144
##  n= 1e+05 일 때 추정된 PI :  3.14316
##  n= 1e+06 일 때 추정된 PI :  3.139088
```