R컴퓨팅

9강 함수 만들기

정보통계학과 장영재 교수

1. 함수의 정의
2. 함수의 기초
3. 여러 가지 함수의 생성

1 함수의 정의

1 함수의 정의

- 함수(function)란 특정한 작업을 독립적으로 수행하는 프로그램 코드의 집합체

R의 내장함수에 사용자가 원하는 특정한 기능이 구현되어 있지 않다면 사용자 스스로 직접 함수를 생성하여 원하는 기능을 수행할 수 있음

2 함수의 기초

2 함수의 기초

1 함수의 장점

- 작업을 작은 단위로 분할하여 수행하도록 함으로써 효율성 제고
- 자유로운 수정·보완이 가능하므로 작업의 유연성이 확보됨
- 작업에 사용하는 코드의 오류 등의 발생 원인 파악이 용이
- 코드의 크기를 줄임으로써 작업 프로세스 이해가 용이
- 사용자가 정의한 함수를 자유롭게 활용하여 작업 목적을 충족

2 함수의 기초

2 함수의 구조 및 생성방법

[1] 함수의 구조

› 함수의 일반적인 구조는 다음과 같음

```
함수이름 <- function(매개변수) {함수의 몸체}
```

› function은 함수를 정의를 위해 사용하는 R의 내장 함수

- list()함수가 list 클래스 객체를 생성하듯이
  function()이 함수 클래스 객체를 생성

2 함수의 기초

2 함수의 구조 및 생성방법

› 함수는 매개변수와 몸체부분으로 나눌 수 있음

- 매개변수라는 것은 일반적으로 함수 선언부나 정의 부분에 있는 변수를 의미
- 중괄호 { }로 묶여 있는 함수의 몸체 부분은 함수가 호출될 때 실행하게 될 내용들

2 함수의 기초

2 함수의 구조 및 생성방법

- 이러한 구조에 따라 함수는 입력, 연산, 출력의 3단계를 거쳐 실행됨
- 입력단계에서는 매개변수를 통해 입력된 자료값을 읽어 들이는 작업을 수행
- 연산단계에서는 입력 데이터를 변환하거나 계산하는 작업을 실시
- 출력단계에서는 연산결과를 객체로 반환하거나 화면에 출력하는 작업

2 함수의 기초

2 함수의 구조 및 생성방법

[2] 변수의 종류

- C 등과 같은 프로그래밍 언어와 마찬가지로 R에서도 변수의 유효범위에 따라 지역변수, 전역변수, 매개변수(인수) 등으로 변수를 구분

- 전역변수는 어떤 변수의 영역 내에서도 접근할 수 있는 변수로서 프로그램이 종료될 때까지 사용 가능한 상태로 존재하다가 프로그램이 종료되면 소멸

2 함수의 기초

2 함수의 구조 및 생성방법

- 지역변수란 전역변수와는 다르게 함수 내부에서만 사용이 가능한 변수로서 함수 내에서 정의된 후 함수가 호출되면 생성되었다가 함수가 종료되면 함께 소멸이 되는 변수

- 지역변수가 전역변수와 동일한 이름을 갖고 있을 경우에는 함수 내에서 지역변수처럼 인식이 되지만, 함수를 벗어나서는 전역변수 값이 그대로 보존되고 함수 내에서의 연산이 영향을 미치지 않음

- 매개변수는 함수의 구조에서 설명하였듯이 함수로 전달되는 값으로 함수를 생성할 때 소괄호 내에 선언해 주며 함수 내에서만 사용 가능

2 함수의 기초

2 함수의 구조 및 생성방법

› 보기 9-1: ① u, v는 전역변수로서 함수 외부에서 정의하여 각각 1, 8 값을 할당 ② 매개변수 x를 받아 x에 1을 더하고 이러한 x 값에 u의 값을 더한 뒤 다시 u에 저장·출력하는 함수를 생성

```
> u <- 1          # 전역변수
> v <- 8          # 전역변수
> g <- function(x) {          # x : 매개변수
+ x <- x + 1
+ u <- u + x # u는 지역변수의 역할
+ return(u)
+ }
```

2 함수의 기초

2 함수의 구조 및 생성방법

```
> g(v)      # v에 8이라는 값이 할당되어 있으므로 g(8)의 의미
[1] 10
> u         # 전역변수는 값의 변화가 없음
[1] 1
> v         # 전역변수는 값의 변화가 없음
[1] 8
```

2 함수의 기초

2 함수의 구조 및 생성방법

[3] 함수의 생성 방법

› R에서 함수를 생성하는 방법은 크게 세 가지로 나뉨

- R 콘솔에서 직접 입력하여 생성하는 방법과 fix 함수를 이용하는 방법, 그리고 외부파일에 저장하여 읽어 들이는 방법

2 함수의 기초

2 함수의 구조 및 생성방법

› 보기 9-2: ① R 콘솔에서 직접 입력하여 작성하는 방식으로 데이터 값을 받아 산술평균을 구하는 d.mean 함수를 작성 ② 평균 4이고 표준편차 1인 정규분포에서 100개의 샘플을 추출한 뒤 이를 x라는 변수에 저장하고 d.mean 함수를 이용하여 산술평균을 산출

```
> d.mean<-function(data) {     # 함수의 선언
+ sum(data) / length(data)    # 매개변수 값으로 들어온 data의 합을 data의
+  # 크기로 나누기
+ }
> x<-rnorm(100, mean=4, sd=1)      # 평균이 4이고 표준편차가 1인
>                                  # 정규분포에서 100개의 샘플을 추출
> d.mean(x)                  # x의 평균을 구하는 함수
[1] 4.147136
```

2 함수의 기초

2 함수의 구조 및 생성방법

› 보기 9-3: fix()함수를 이용하여 매개변수 data를 받아서 분산을 구하는 함수 만들기

```
〉 fix(d.var)

〉 function (data)    # 함수의 선언
+ {
+ data.var 〈– sum((data – d.mean(data))^2) / (length(data)–1)  # 분산 공식
+ return(data.var)              # data.var 값을 반환
+ }
```

2 함수의 기초

2 함수의 구조 및 생성방법

- 보기 9-4: 데이터 값을 받아서 데이터의 최솟값과 최댓값을 각각 data.min, data.max변수에 할당하고 data의 범위를 출력하는 함수 만들기

- 메모장을 열어서 다음과 같이 입력하고 rangefunction.txt 로 저장

2 함수의 기초

2 함수의 구조 및 생성방법

```
> d.range <- function(data)          # 함수의 선언
+ {
+ data.min<-min(data)          # 데이터의 최솟값 산출
+ data.max<-max(data)          # 데이터의 최댓값 산출
+ c(data.min, data.max)        # 데이터의 범위를 출력
+ }
> source("C:₩rangefucntion.txt")        # 함수의 생성
> d.range(x)            # 보기 9-2에서 생성하였던 x의 범위 출력
[1] 1.404092 6.991365
```

2 함수의 기초

2 함수의 구조 및 생성방법

[4] 함수의 편집

- 경우에 따라 생성한 함수가 오류가 있다든지 작업을 효율적으로 수행하기 위해서 함수를 편집할 필요

- fix() 함수나 edit() 함수를 사용하면 편집창이 열리게 되므로 이를 이용하여 함수를 편집

3 여러 가지 함수의 생성

3 여러 가지 함수의 생성

1 함수의 저장

- 보기 9-5: ① x와 y라는 데이터 값을 받아서 두 수의 합을 계산하는 함수 f1과 차를 계산하는 f2를 생성 ② 함수 f1을 f라는 이름으로 지정하여 f(3,2)를 계산하고 f2를 f라는 이름으로 지정하여 f(3,2)를 계산하기

```
> f1 <- function(x,y) {return(x+y)}
> f2 <- function(x,y) {return(x-y)}
> f <- f1        # f1을 f라는 이름으로 저장
> f(3,2)
[1] 5
> f <- f2        # f2를 f라는 이름으로 저장
> f(3,2)
[1] 1
```

3 여러 가지 함수의 생성

1 함수의 저장

› 보기 9-6: 보기 9-4에서 생성한 f1와 f2 및 x와 y를 매개변수로 하는 함수 g를 생성하여 x와 y의 합과 차를 계산하는 작업을 수행

```
〉 g 〈– function(h,x,y) {h(x,y)}      # h는 함수를 받는 매개변수
〉 g(f1,3,2)                          # h는 f1, x=3, y=2
[1] 5
〉 g(f2,3,2)                          # h는 f2, x=3, y=2
[1] 1
```

3 여러 가지 함수의 생성

2 함수의 매개변수 지정

› 보기 9-7: x에 (1,2,3,4)의 값을, y에 (4,3,2,1) 값을 저장하고 x+y의 값을 z에 저장한 뒤 z 값을 출력

```
> f0<-function() {        # 매개변수가 없는 함수 선언
+ x<-c(1,2,3,4)
+ y<-c(4,3,2,1)
+ z<-x+y
+ print(z)               # z를 출력
+ }
> f0()
[1] 5 5 5 5
```

- 위의 예서 주의할 점은 함수 선언에 관련된 부분으로 매개변수가 없더라도 function()과 같이 반드시 괄호를 표시하도록 할 것

3 여러 가지 함수의 생성

2 함수의 매개변수 지정

› 보기 9-8: data와 num이라는 매개변수를 받아 data의 평균과, 분산, 범위를 구하는 함수를 생성(num이라는 매개변수에 따라 평균, 분산, 범위 중 한 가지 통계량만을 구하여 출력하되 기본값은 평균 출력)

```
> f_default<-function(data, num=1) {    # "num=1"은 기본 값으로 1
+ d.min<-min(data)
+ d.max<-max(data)
+ switch(num, mean(data), var(data), c(d.min, d.max)) # switch 함수
+ }
```

3 여러 가지 함수의 생성

2 함수의 매개변수 지정

› 보기 9-9: 평균이 5이고 표준편차 2인 정규분포를 따르는 난수를 1000개 발생시켜 x에 저장한 뒤 보기 9-8에서 정의한 함수 f_default를 사용하여 x의 분산을 구해 보자. 또한 이 값과 num값을 주지 않았을 때의 결과 값을 비교해 보자.

```
> x<-rnorm(1000, mean=5, sd=2)        # random number를 생성
>                                     # normal 정규분포를 따름 (평균 5, 표준편차 2)
> f_default(data=x, num=2)
[1] 3.937556
> f_default(x, 2)
[1] 3.937556
> f_default(x)
[1] 5.150254
```

3 여러 가지 함수의 생성

3 함수를 위한 함수

[1] is.function()

- is.function() 함수는 객체가 함수인지 아닌지 검증하는 함수

- 지정한 객체가 함수라고 판단되면 TRUE값을, 함수가 아니면 FALSE 값을 반환

3 여러 가지 함수의 생성

3 함수를 위한 함수

- 보기 9-10: 보기 5-8과 같이 f_default라는 함수를 정의하고, 이렇게 생성된 f_default라는 객체가 함수인지를 판별

```
> f_default<-function(data, num=1) {
+ d.min<-min(data)
+ d.max<-max(data)
+ switch(num, mean(data), var(data), c(d.min, d.max))
+ }
> is.function(f_default)
[1] TRUE
```

- is.function(f_default) 명령문을 실행하였더니 TRUE 값이 반환되었으므로 f_default라는 객체는 함수임을 알 수 있음

3 여러 가지 함수의 생성

3 함수를 위한 함수

[2] args()

- args() 함수는 함수의 매개변수들을 반환하는 함수

- 보기 9-11: 보기 9-8의 f_default라는 함수의 매개변수로 무엇을 사용하는지 출력하기

```
> args(f_default)
function (data, num = 1)
NULL
```

- data 매개변수와 num 매개변수를 갖고 있으며 num 매개변수는 기본 값으로 1을 가진다는 것을 알 수 있음

3 여러 가지 함수의 생성

3 함수를 위한 함수

- 보기 9-12: R에 내장된 기본함수의 경우에도 args() 함수를 이용하면 매개변수를 파악할 수 있음

```
> args(log)
function (x, base = exp(1))
NULL
```

- x 매개변수와 base 매개변수를 갖고 있으며 base 매개변수의 기본값은 exp(1)의 값임을 나타내고 있음

3 여러 가지 함수의 생성

3 함수를 위한 함수

[3] attributes()

- attributes() 함수는 함수의 소스코드를 반환하는 역할을 하므로 R에서 사용자가 직접 생성한 함수가 인식이 되었다면 그 소스를 attributes() 함수를 통해 확인해 볼 수 있음

3 여러 가지 함수의 생성

3 함수를 위한 함수

- 보기 9-13: attributes() 함수를 이용하여 보기 9-8의 f_default라는 함수의 소스를 출력하기

```
> attributes(f_default)
$srcref
function(data, num=1) {
d.min<-min(data)
d.max<-max(data)
switch(num, mean(data), var(data), c(d.min, d.max))
}
```

- attribute() 함수를 실행하게 되면 $srcref라는 이름 아래에 저장되어 있던 소스코드가 나타나게 됨

3 여러 가지 함수의 생성

4 함수 생성방법의 응용

[1] 연산자의 생성

› 사용자의 정의에 따라 연산자를 생성하고 싶다면 %로 시작과 끝을 이루는 함수명을 갖는 함수를 생성

› 보기 9-14: 두 변수 값을 받아서 앞의 변수값과 뒤쪽 변수값의 두 배를 더하는 연산자를 생성

```
〉 "%a2b%" 〈– function(a,b) return(a+2*b)
〉 3 %a2b% 5
[1] 13
```

- 주의할 점은 연산자 생성 시 반드시 인용부호와 % 표시를 병기해야 한다는 것

3 여러 가지 함수의 생성

4 함수 생성방법의 응용

[2] 함수 내에서의 함수 생성

- 함수는 객체이므로 함수 내에서 다른 함수를 정의할 수 있음

- 보기 9-15: 아래 함수 f는 v라는 지역변수에 1이라는 값을 저장하고 이 함수 매개변수 y 값을 받은 뒤, 이 y 값과 v, 그리고 외부에서 지정되는 u 값을 더하여 제곱연산을 실시하는 함수

3 여러 가지 함수의 생성

4 함수 생성방법의 응용

```
> f <- function(x) {
+ v<-1
+ g<-function(y)
+ return((u+v+y)^2)
+ gu<-g(u)
+ print(gu)
+ }
> u<-6
> f()
[1] 169
```

R컴퓨팅