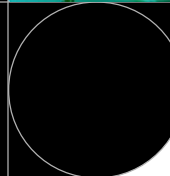
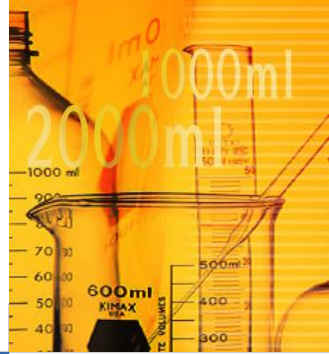


Chapter 2

R 기초 (1) Vector

Sejong Oh

Bio Information technology Lab.



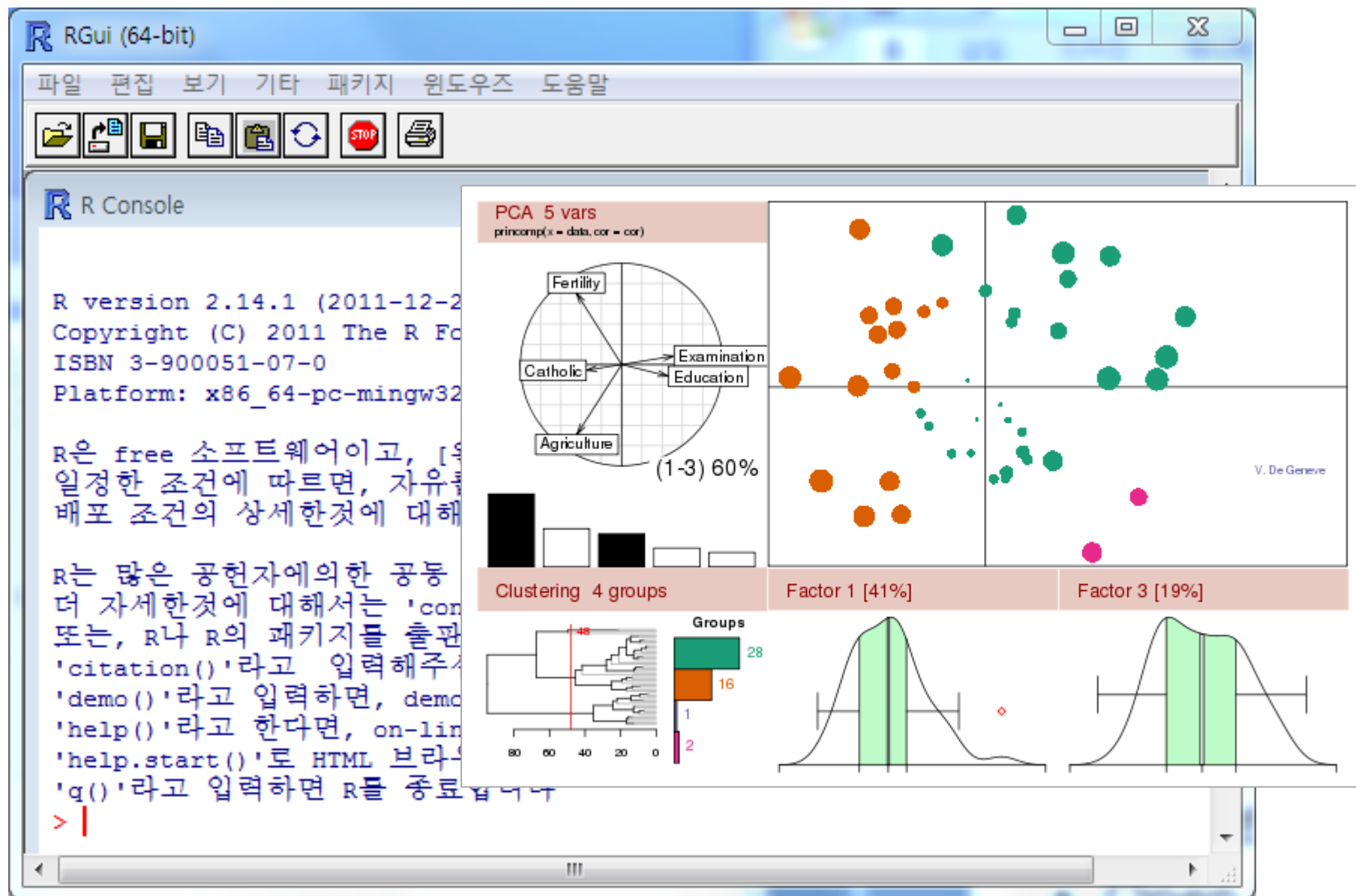
Content

- R 이란?
- R 기본사용
- 벡터(vector)
- 벡터 연산과 함수
- R 사용 Tip

R 이란?

- 자료분석, 통계작업을 지원하는 공개 SW
- R 은 계산기
- R 은 프로그래밍 언어
- 자료, 통계 분석을 위한 거의 모든 기능을 함수, 패키지 형태로 제공
- Java, C 프로그램과 연동 가능
- Compact size (< 60 Mb)
- Open Source

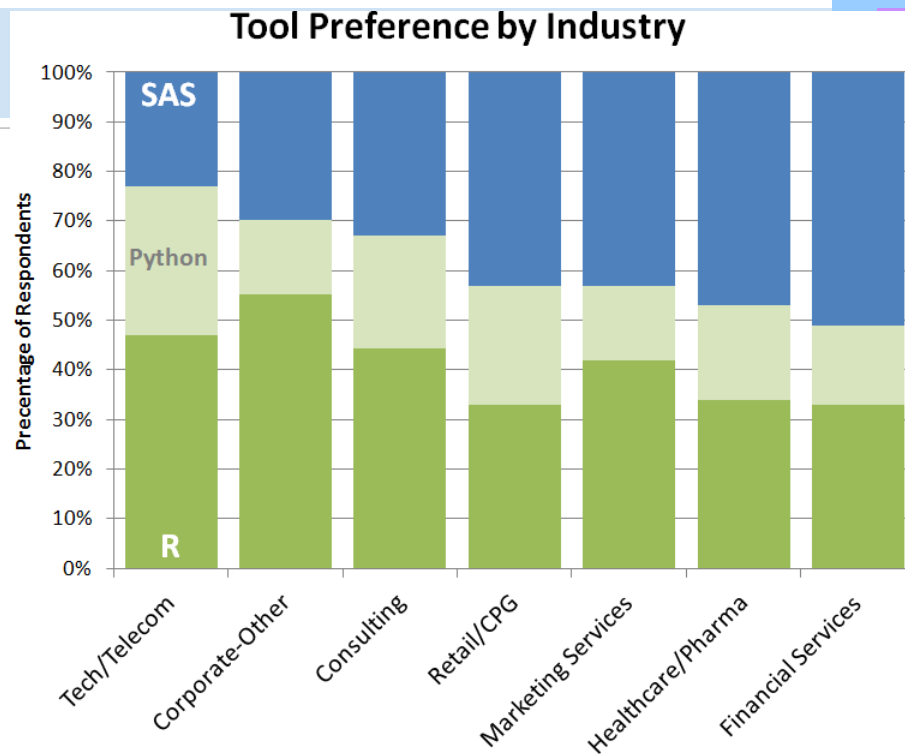
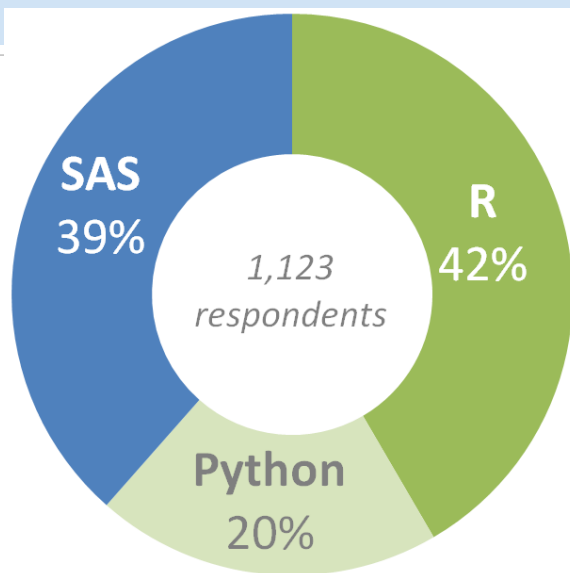
R 이란?



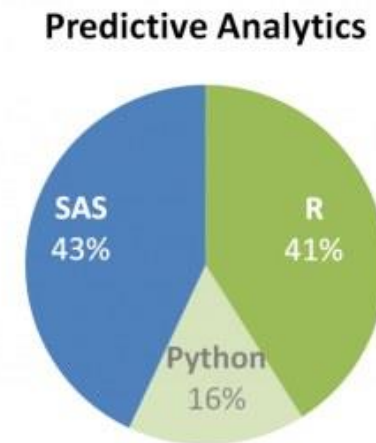
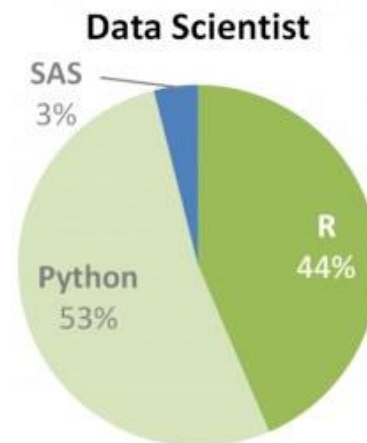
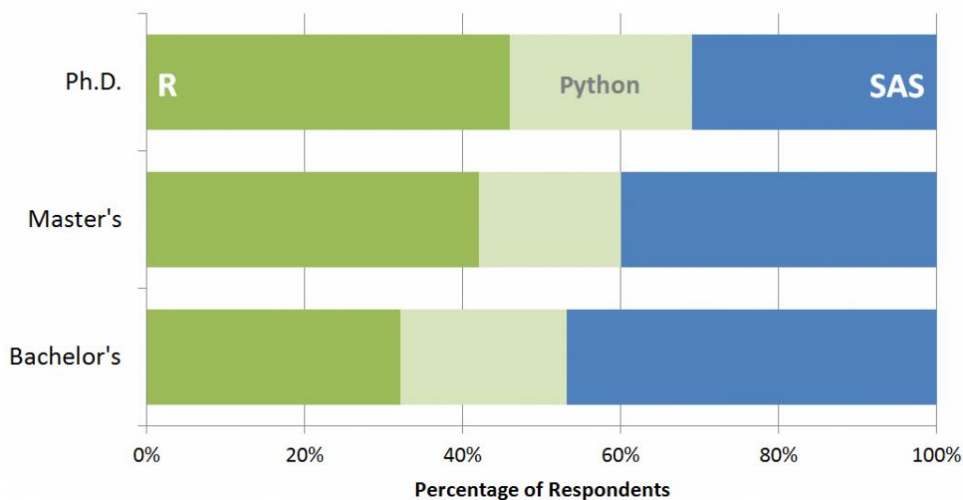
- R
 - ◉ 쉬운 문법 구조
 - ◉ 데이터 분석을 목적으로 개발
 - ◉ 방대한 패키지, 최신 이론이 즉시 패키지로 제공
 - ◉ 뛰어난 데이터 시각화
 - ◉ 폭 넓은 커뮤니티
 - ◉ 데이터 사이언스의 공용어로 인식됨
 - ◉ R 은 C, Java 와 같은 프로그래밍 언어가 아님
 - ◉ 느린 속도

- Python

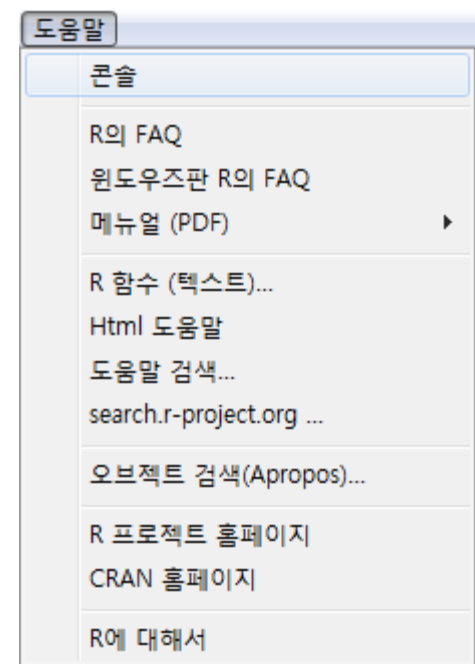
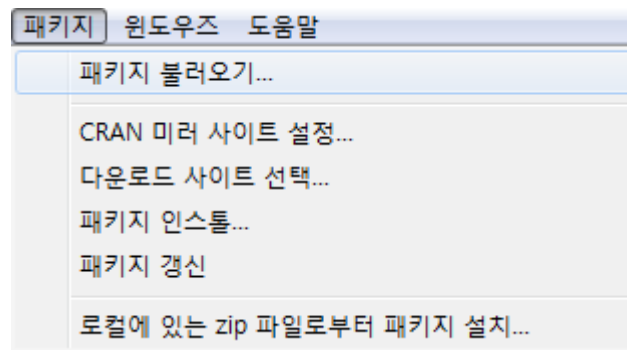
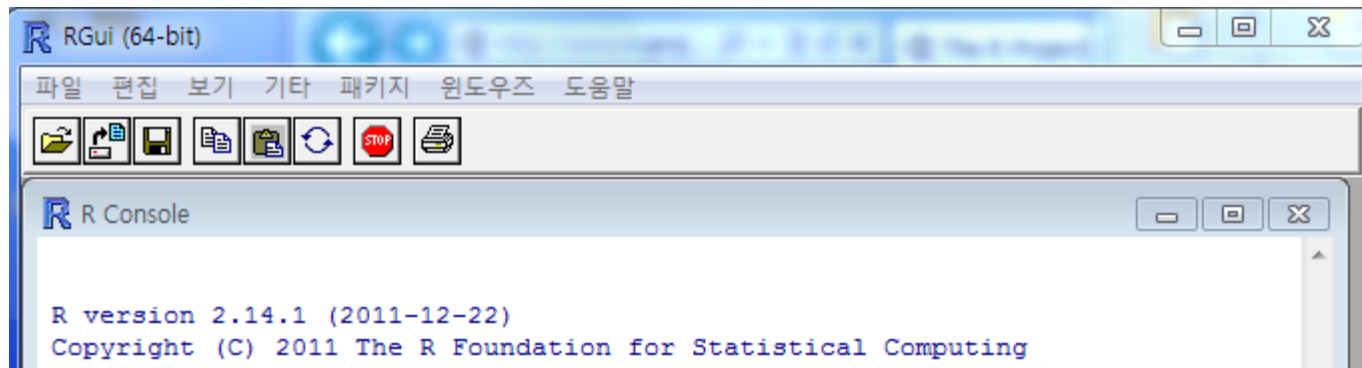
- 데이터 분석 뿐만 아니라 다양한 SW 개발 분야에 활용
- 완전한 프로그래밍 언어
- 딥러닝 패키지에 대한 인터페이스
- 시각화 기능이 R 보다 떨어짐



Tool Preference by Education



- R user interface



● 계산기로 사용하기

```
>2+3  
>(3+6)*8  
>2^3
```

2의 세제곱을 계산함

- 사칙연산자 : +, -, *, /, ^
- 나머지 : %%
- 주석문 (comment) : #

● 함수 사용하기

```
>log(10)+5
```

로그함수

```
>sqrt(25)
```

제곱근

```
>max(5,3)
```

두 값중 큰 값

- log(), sqrt(), max(), min(), ...

[연습 1]

- R 을 사용하여 다음의 계산식에 대한 해답을 구하시오

$$25+99$$

$$456 - 123$$

$$2 \times (3+4)$$

$$(3+5 \times 6) \div 7$$

$$(7 - 4) \times 3$$

$$2^{10} + 3^5$$

1256 을 7 로 나눈 나머지

184 를 5 로 나눈 나머지

$$1976 \div 24$$

$$16 \times 25 + 186 \times 5 - 67 \times 22$$

- 변수(variable) 사용하기

```
a <- 10
b <- 20
c <- a+b
c
```

c 에 저장된 값을 출력하라는 의미

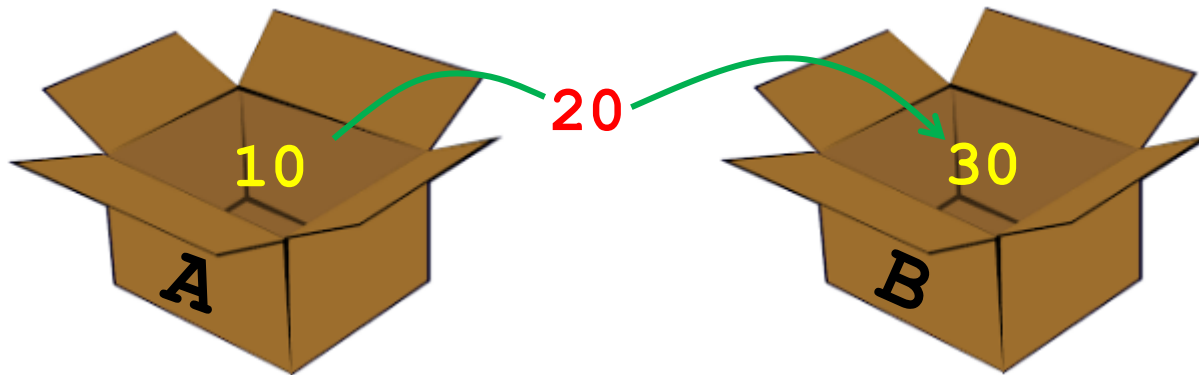
- 프로그래밍 언어의 변수와 유사함
- 변수의 자료형(data type)은 지정하지 않는다
- 올바른 자료가 저장되었는를 검사하지 않으므로 주의
- 문자형 자료의 저장 : "" 이용

```
a <- 10 # a는 숫자저장 변수
b <- 20
a+b
a <- "A" # a는 문자저장 변수
a+b # 에러발생
d<-10; e<-15; f<-20 # 한줄에 여러 명령문을 입력
```

- 변수(variable)란

- 어떤 값(숫자, 문자, True/False 등)을 임시로 보관해 놓기 위한 저장소
- 저장된 값이 바뀔 수 있다는 의미에서 변수라고 함

```
A <- 10  
B <- A+20
```



- 변수이름 규칙

- 첫글자는 문자나 . (dot) 으로 시작
- 그 이후에는 문자, 숫자, dot, underline 사용 가능
- 대소문자를 구분한다.

```
avg <- 10  
AVG <- 20  
val.a <- 15  
val.b <- 20  
val.A <- 19
```

- 변수에 값을 할당

- '<-' 또는 '=' 사용

```
var2 <- 15  
var1 = 10
```

- Note1. 한번 만들어 사용한 변수는 R 을 종료할 때 까지 사라지지 않는다

```
avg <- 10
AVG <- 20
val.a <- 15
val.b <- 20
val.A <- 19
...
print(avg)
```

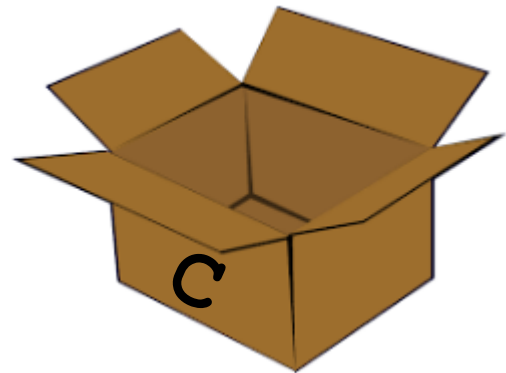
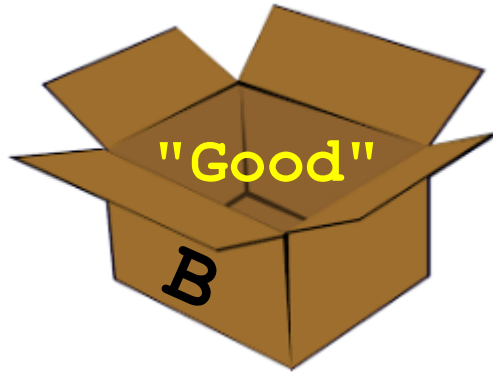
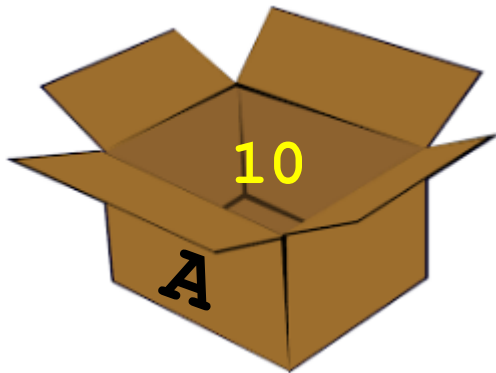
- Note2. 하나의 변수는 다양한 유형의 값을 저장할 수 있다

```
V1 <- 10
V1
V1 <- "Good Morning"
V1
```

- R 에서 사용할 수 있는 자료형(data type)
 - 숫자 : 1, 456, -123, 2.15
 - 문자 : "a", "b", "c", "hello", "good"
 - 논리형 : TRUE, FALSE (반드시 대문자)
 - 특수한 값
 - NULL : 비어있는 값. 자료형도 없고 길이도 0
 - NA : 결측값 (missing value)
 - NaN : 수학적으로 정의가 불가능한 값 (예 : `sqrt(-3)`)
 - Inf, -inf : 양의 무한대, 음의 무한대



```
A <- 10  
B <- "Good"  
C <- NULL
```



[연습 2]

- (문제1) 1차식 $y = 3x + 4$ 를 R을 이용하여 해결하고자 한다.
- x 가 5, 6, 8 일 때 y 값을 각각 구하시오.

(해답)

```
x <- 5
y <- 3 * x + 4
y
x <- 6
y <- 3 * x + 4
y
x <- 8
y <- 3 * x + 4
y
```

[연습 2]

- (문제2) 원의 넓이를 구하는 식을 R 을 이용하여 구하고자 한다
- 반지름이 10,15,20 일 때 원의 넓이를 각각 구하시오

- (문제3) 2차식 $y = 2x^2 + 5x + 10$ 에 대해 x 가 6,8,10 일 때 y의 값을 각각 구하시오

벡터

- 우리가 분석하고자 하는 데이터는 대부분 1차원 배열 또는 2차원 배열의 형태를 가지고 있다

- 1차원 배열 데이터

- 1학년 학생들의 몸무게 자료
- 2학년 학생들의 영어성적
- 1학년 학생들의 선호하는 색깔 자료

56	60	72	64	80	55	59	69	70
----	----	----	----	----	----	----	----	----

- 2차원 배열 데이터

- 3학년 학생들의 전과목 성적 자료

92	2	1	1	2	0
36	2	2	1	1	0
105	3	2	1	1	0
81	1	2	1	1	0
94	1	1	2	1	0
20	1	1	3	3	0
50	1	2	1	1	0
68	3	3	2	1	0
89	3	1	3	2	0
19	3	2	1	3	0
118	2	1	2	1	0

벡터

- 1차원 데이터를 저장하기 위한 자료 구조를 R에서는 벡터(vector)라고 한다
- 수학적 의미의 vector 와 다루는 방법이 동일
- 변수 : 하나의 값만 저장 가능
- 벡터 : 동일한 형태의 값을 여러 개 저장 가능

A



B



- 데이터 벡터(data vector)

```
x <- c(1,2,3)
y <- c(4,5)
z <- c("a","b","c")
x
y          # y 에 저장된 값을 출력하라는 의미
```

- c() 함수를 이용해서 생성
- 데이터 벡터는 문자, 숫자중 한가지 타입만 가능
- 데이터 벡터에 문자, 숫자를 섞어서 넣으면 모두 문자형으로 인식

```
w <- c(1,2,3, "a","b","c")
```

```
> w <- c(1,2,3, "a","b","c")
> w
[1] "1" "2" "3" "a" "b" "c"
```

```
x <- c(1, 2, 3)  
z <- c("a", "b", "c")
```

x

1	2	3
---	---	---

z

"a"	"b"	"c"
-----	-----	-----

- 연속적인 숫자로 이루어진 벡터 만들기

```
v1 <- 50:90  
v1
```

```
> v1 <- 50:90  
> v1  
[1] 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73  
[25] 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90  
>
```

```
v2 <- c(1,2,5, 50:90)  
v2
```

```
> v2 <- c(1,2,5, 50:90)  
> v2  
[1] 1 2 5 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67  
[22] 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88  
[43] 89 90  
>
```

- 데이터 벡터(data vector)
 - 데이터 벡터는 구성 값에 이름 부여 가능
 - names() 함수 이용

```
score <- c(90,85,70)      # 성적
names(score) <- c("John", "Tom", "Jane")
score                     # 이름과 함께 값이 출력
```

```
> score <- c(90,85,70)
> names(score) <- c("John", "Tom", "Jane")
> score
John  Tom  Jane
  90   85   70
>
```


- 데이터 벡터에서 값 추출하기
 - 한 개의 값 추출

```
d <- c(1, 4, 3, 7, 8)  
d[2] # 4 가 출력 (R 은 인덱스를 1부터 시작)
```

d	1	4	3	7	8
	d[1]	d[2]	d[3]	d[4]	d[5]

- 데이터 벡터에서 값 추출하기

- 구간의 값 추출

```
d <- c(1,4,3,7,8)
d[1:3]           # 처음 세개의 자료 출력
d[c(1,3,5)]      # 홀수번째 자료 출력
```

- Negative index

```
d <- c(1,4,3,7,8,9)
d[-2]            # - 는 '제외하고' 의 의미.
d[-c(3:5)]       # 세번째에서 다섯번째 값은 제외하고
```

- 데이터 벡터에서 값 추출하기
 - 이름으로 값 추출하기

```
GNP <- c(2090, 2450, 960) # GNP
names(GNP) <- c("Korea", "Japan", "Nepal")
GNP[1]
GNP["Korea"]
GNP[c("Korea", "Nepal")]
```

```
> GNP <- c(2090, 2450, 960) # GNP
> names(GNP) <- c("Korea", "Japan", "Nepal")
> GNP[1]
Korea
 2090
> GNP["Korea"]
Korea
 2090
> GNP[c("Korea", "Nepal")]
Korea Nepal
 2090   960
```

[연습3]

- 101 ~ 200 의 값으로 구성된 벡터 d 를 생성하시오

```
d <- 101:200
```

- d 에 어떤 값이 저장되었는지 확인하시오
- d 에서 10번째 값은 무엇인가
- d 에서 뒤에서 10개의 값을 잘라내어 보이시오
- d 에서 앞에서 20 개의 값을 잘라내어 d.20 에 저장하시오. d.20 의 값을 보이시오
- d.20 에서 5번째 값을 제외한 나머지 값들을 보이시오
- d.20 에서 5,7,9 번째 값을 제외한 나머지 값을 보이시오

힘내자!



- 벡터에 대한 산술 연산

```
d <- c(1,4,3,7,8)
2*d
d-5
3*d + 4
```

```
> d <- c(1,4,3,7,8)
> 2*d
[1] 2 8 6 14 16
> d-5
[1] -4 -1 -2 2 3
> 3*d + 4
[1] 7 16 13 25 28
>
```

- 데이터 벡터간 연산

- 두 벡터의 연결

```
x <- c(1,2,3)
y <- c(4,5)
c(x,y)          # 단순히 x,y 를 연결하여 출력
z <- c(x,y)     # x,y 를 연결하여 z에 저장
```

- 두 벡터의 합 (두 벡터의 길이가 같아야 가능)

```
x <- c(1,2,3)
y <- c(4,5,6)
x+y             # 대응하는 원소끼리 + 하여 출력
z <- x + y      # x,y 를 더하여 z에 저장
```

(사칙연산 모두 적용가능)

- 데이터 벡터에 적용 가능한 함수들

함수명	설명
sum()	자료의 합
mean()	자료의 평균
max(), min()	자료의 최대, 최소값
var()	자료의 분산 값
sort()	자료를 정렬하여 출력
range()	자료의 범위 (최대값 ~ 최소값)
length()	자료의 개수
cumsum()	누적합계
diff()	인접값과의 차이

- 데이터 벡터에 적용 가능한 함수들

```
d <- c(1,2,3,4,5,6,7,8,9,10)
sum(d)
sum(2*d)
length(d)
mean(d[1:5])
max(d)
min(d)
sort(d) # 올림차순 정렬
sort(d, decreasing = FALSE) # 올림차순 정렬
sort(d, decreasing = TRUE) # 내림차순 정렬
```


[연습4]

- d1, d2 가 다음과 같을 때 질문에 답하시오

```
d1 <- 1:50  
d2 <- 51:100
```

- d1, d2 의 값을 보이시오
- d1+d2, d2-d1, d1*d2, d2/d1 의 결과를 각각 보이시오
- d1, d2 의 값들의 합(sum)을 각각 보이시오
- d1, d2 에 있는 모든 값들의 합(sum)을 보이시오
- d2 에서 가장 큰 값과 가장 작은 값을 보이시오
- d2 와 d1 의 값들의 평균값을 각각 구하고 두 평균의 차이를 보이시오
- d1 의 값들을 큰수에서 작은 수 순으로 정렬하여 보이시오
- d1 과 d2 에서 큰수 순으로 각각 10개씩을 추출하여 d3 에 저장하시오 (결과적으로 d3 에는 20개의 수가 저장)



- 사용 Tip

- 함수의 사용법 알고 싶을 때 (함수 이름을 알면)

```
help(sum)                # help("sum") 도 가능  
? sum
```

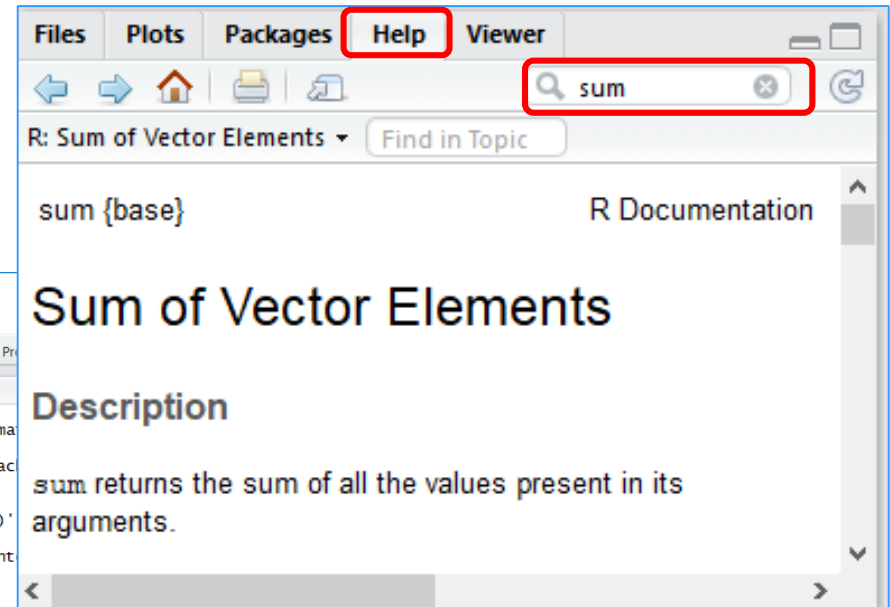
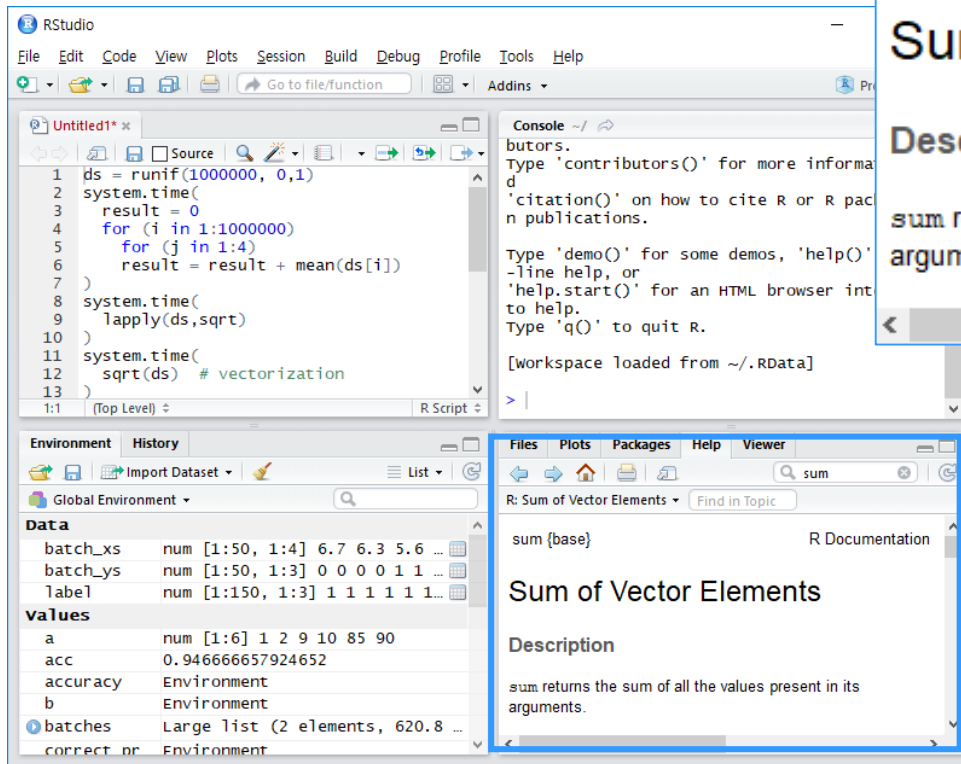
- 함수의 사용법 알고 싶을 때 (함수 이름을 모르면)

```
help.search("average")
```

- History

- R은 최근에 사용한 명령어를 25개까지 기억
- 위아래 화살표 키를 이용해서 이전에 사용한 명령어를 불러올 수 있다
- history() 함수를 이용하여 25개의 목록을 한눈에 확인 가능

- Rstudio 를 이용한 help



```
sum {base}
```

Package 이름

Sum of Vector Elements

Description

`sum` returns the sum of all the values present in its arguments.

Usage

```
sum(..., na.rm = FALSE)
```

Arguments ← 함수의 parameter 값 설명

...
numeric or complex or logical vectors.

`na.rm`
logical. Should missing values (including `NaN`) be removed?

Details ← 함수에 대한 상세 설명

This is a generic function: methods can be defined for it directly or via the [Summary](#) group generic. For this to work properly, the arguments ... should be unnamed, and dispatch is on the first argument.

If `na.rm` is `FALSE` an `NA` or `NaN` value in any of the arguments will cause a value of `NA` or `NaN` to be returned, otherwise `NA` and `NaN` values are ignored.

Logical true values are regarded as one, false values as zero. For historical reasons, `NULL` is accepted and treated as if it were `integer(0)`.

Loss of accuracy can occur when summing values of different signs: this can even occur for sufficiently long integer inputs if the partial sums would cause integer overflow. Where possible extended-precision accumulators are used, but this is platform-dependent.

Value ← 함수의 return 값

The sum. If all of ... are of type integer or logical, then the sum is integer, and in that case the result will be NA (with a warning) if integer overflow occurs. Otherwise it is a length-one numeric or complex vector.

NB: the sum of an empty set is zero, by definition.

S4 methods

This is part of the S4 [Summary](#) group generic. Methods for it must use the signature `x, ..., na.rm`.

'[plotmath](#)' for the use of `sum` in plot annotation.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

[colSums](#) for row and column sums.

Examples ← 함수의 사용 예제

```
## Pass a vector to sum, and it will add the elements together.
sum(1:5)

## Pass several numbers to sum, and it also adds the elements.
sum(1, 2, 3, 4, 5)

## In fact, you can pass vectors into several arguments, and everything gets added.
sum(1:2, 3:5)

## If there are missing values, the sum is unknown, i.e., also missing, ....
sum(1:5, NA)
## ... unless we exclude missing values explicitly:
sum(1:5, NA, na.rm = TRUE)
```


- R package
 - 사용자들이 많이 이용하는 함수들을 미리 구현하여 묶어 놓은 것. 비슷한 함수들을 동일 package 에 모아 놓았다.
 - R 에서 어떤 함수를 사용하기 위해서는 그 함수가 들어 있는 package 를 불러 와야 한다.
 - base 패키지는 가장 기본 적인 함수들을 모아 놓은 package 로서 R 이 구동될 때 자동으로 불러오기 때문에 사용자가 불러 올 필요가 없다.
 - 필요한 package 를 install 하고 불러오는 방법은 다음 장에서 연습



- install한 패키지의 내용을 알고 싶을 때 (포함된 함수 목록, 설명)

```
help(package="class")
```

Functions for Classification



⏪ ⏩

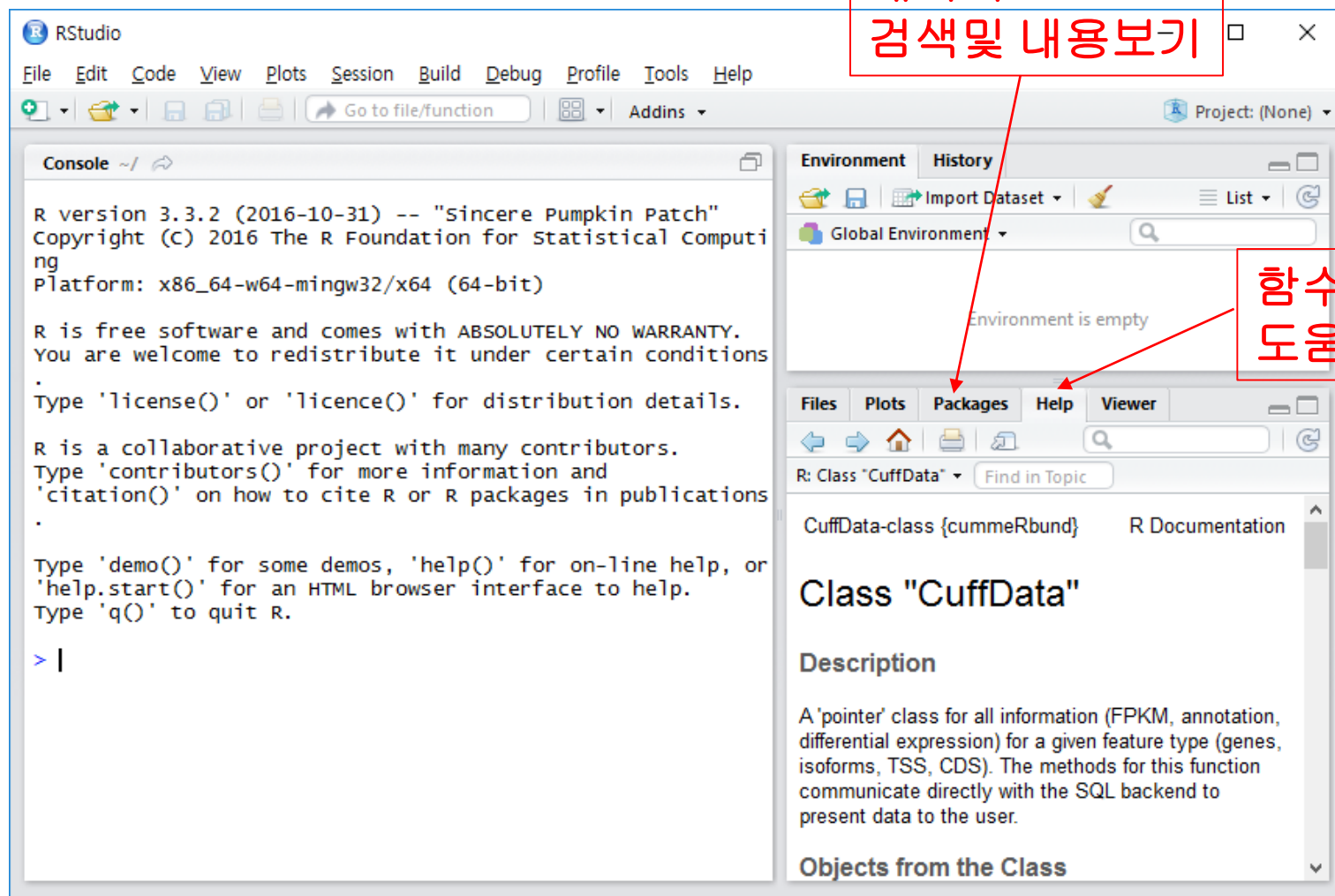
Documentation for package 'class' version 7.3-3

- [DESCRIPTION file.](#)
- [Package NEWS.](#)

Help Pages

batchSOM	Self-Organizing Maps: Batch Algorithm
condense	Condense training set for k-NN classifier
knn	k-Nearest Neighbour Classification
knn.cv	k-Nearest Neighbour Cross-Validatory Classification
knn1	1-nearest neighbour classification
lvq1	Learning Vector Quantization 1
lvq2	Learning Vector Quantization 2.1
lvq3	Learning Vector Quantization 3
lvqinit	Initialize a LVQ Codebook

- Rstudio



[연습5]

- base 패키지가 제공하는 함수들에는 어떤 것들이 있는지 목록을 보이시오
- seq 함수는 어떤 기능을 하는 함수인지 알아보시오.
- Help 페이지에 있는 seq 함수 예제를 실행해 보시오
- seq 함수를 이용하여 1~100 사이의 짝수값을 얻은 뒤 이를 even 에 저장하시오. even 의 값을 보이시오.

