

SOFTWARE PLATFORM

(Big Data – R 入門)



산업 트렌드의 변화와 빅 데이터

세계는 정치, 경제, 사회문화, 예술과 과학 모든 분야에서 급변하고 있다. 특히, 최근의 제4차 산업혁명은 비즈니스와 학문은 물론 우리 생활에도 많은 영향을 주고 있다. 이 근간에는 거대한 데이터의 흐름이 있다. 우리는 이를 빅데이터라고 한다. 빅데이터는 우리 생활과 기업활동에 어떤 영향을 미치고 있을까? 왜 데이터 분석이 중요할까?



1차 산업혁명

기계화
증기기관



2차 산업혁명

대량생산
전기
베세머 전로
컨베이어 벨트



3차 산업혁명

생산의 자동화
애니악



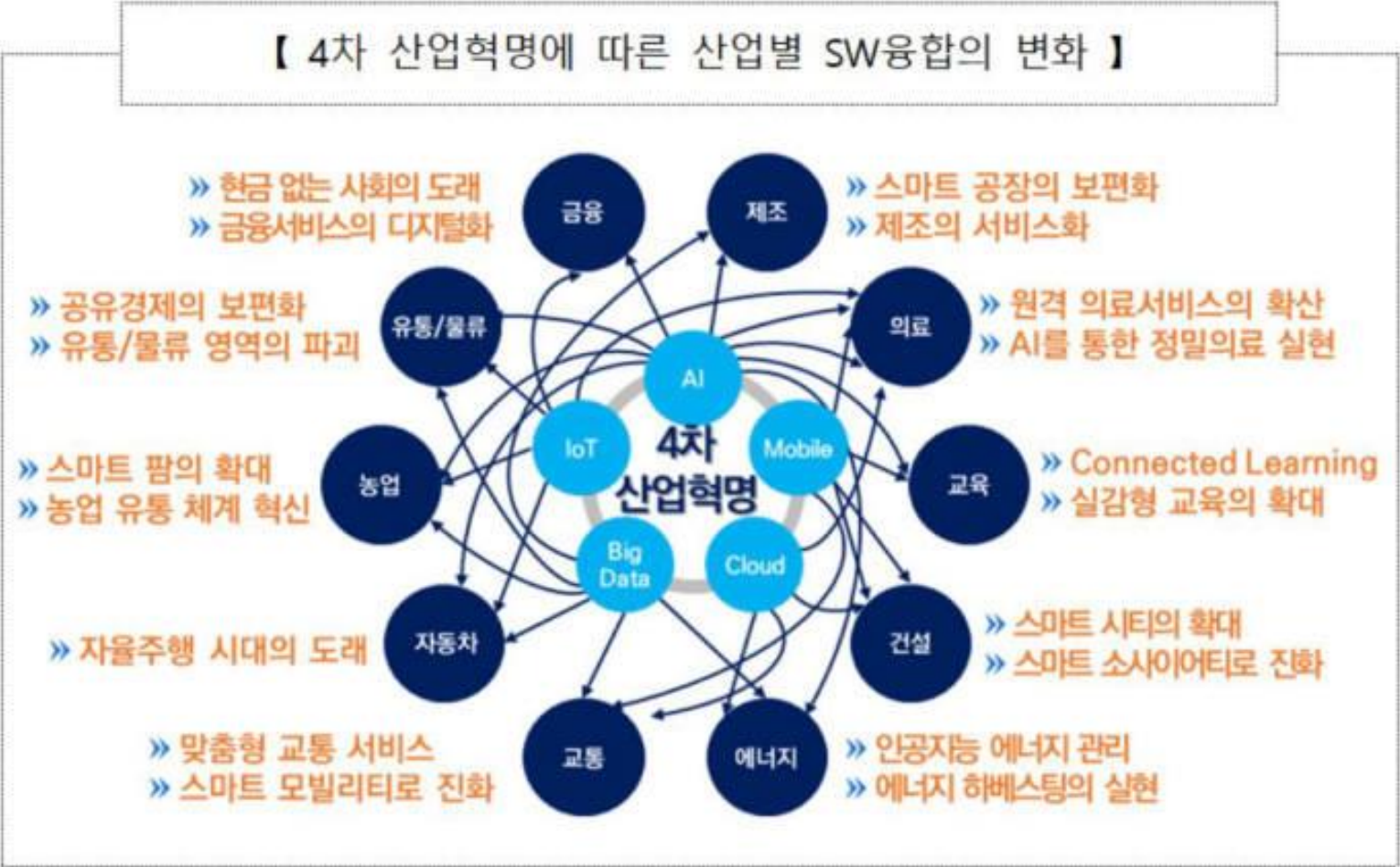
4차 산업혁명

융합/초연결
빅데이터, 인공지능

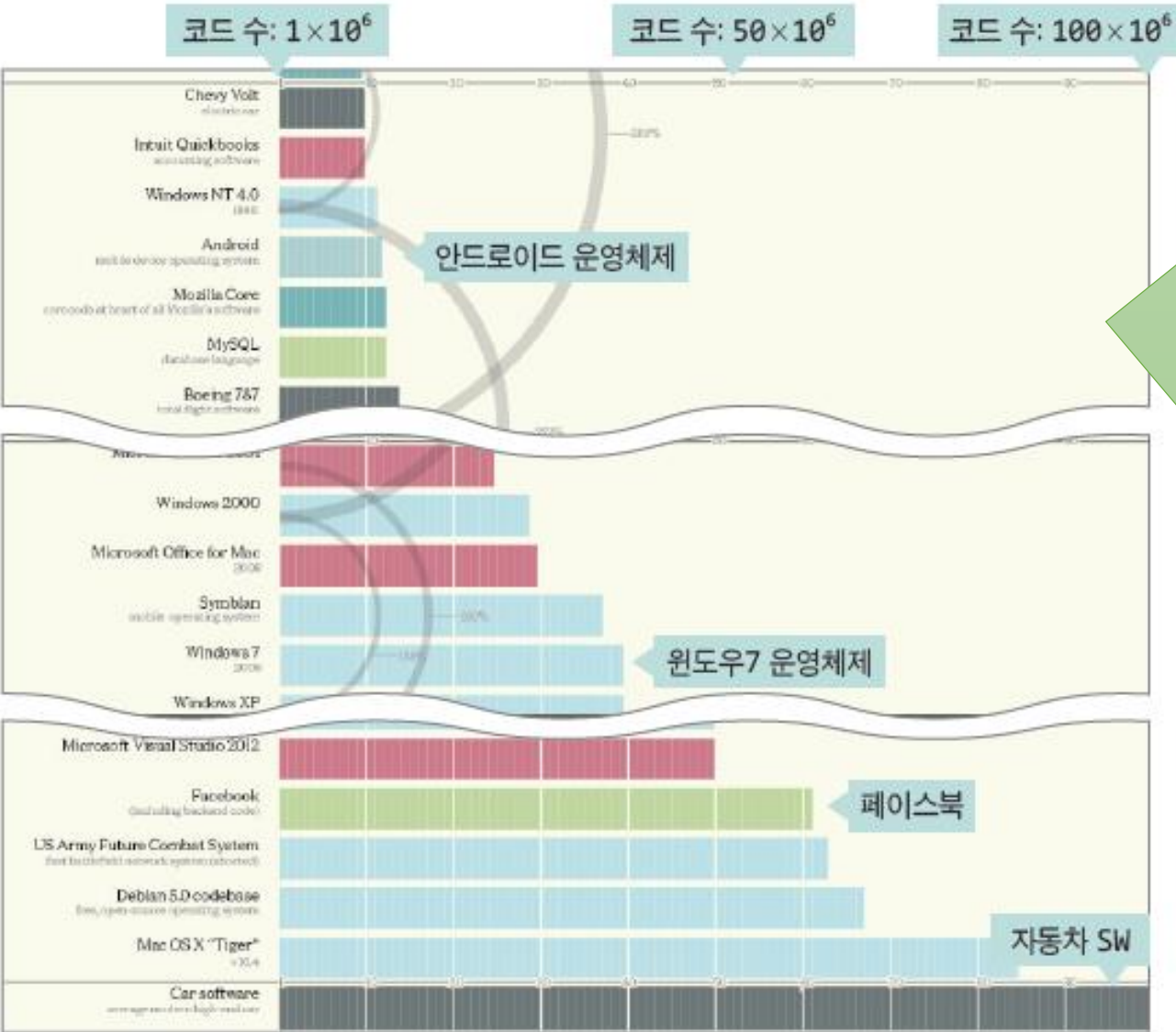


소프트웨어 중심사회

소프트웨어(SW)가 혁신과 성장, 가치창출의 중심이 되고 개인·기업·국가의 경쟁력을 좌우하는 사회



산업별 코드 수의 비교



■ 스마트폰 안드로이드 운영체제에 비해 PC에서 사용되는 윈도우7의 소프트웨어 규모가 훨씬 크다.

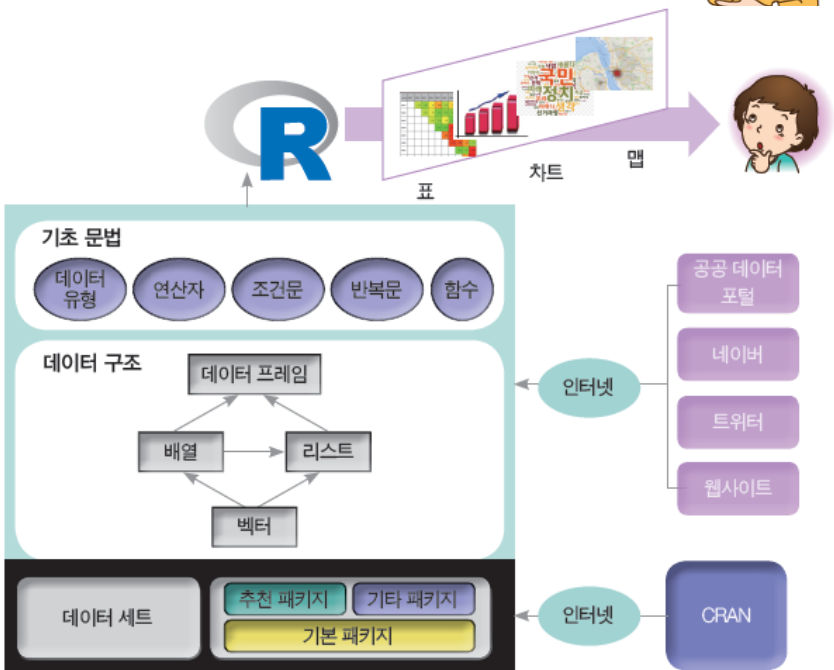
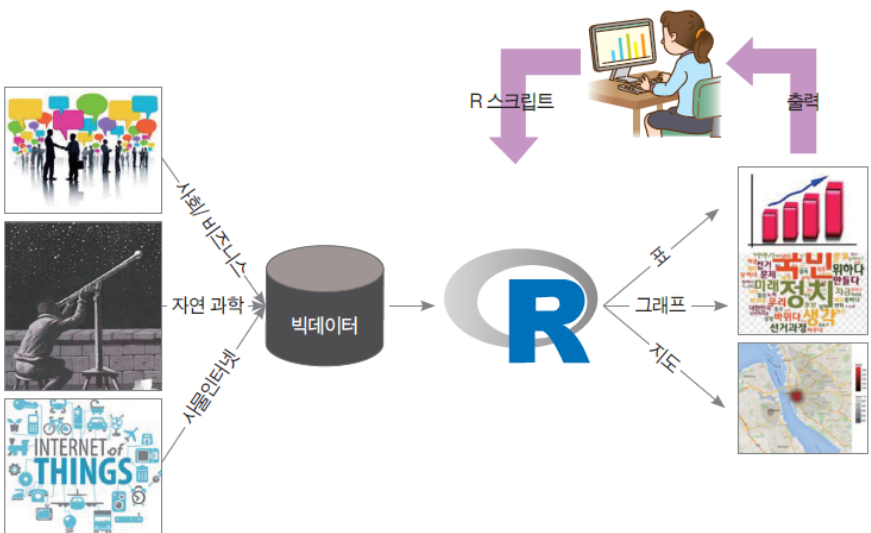
■ 페이스북은 사용자용과 시스템 관리용을 모두 포함하면 PC형 운영체제보다 큰 편이며, 자동차에 사용되는 소프트웨어 규모는 안드로이드 운영체제의 거의 10배에 이른다. 바야흐로 자동차 산업도 하드웨어에서 소프트웨어로 변화하고 있는 추세이다.

코드 수 비교



Why R?

- **R** : 1996년, 뉴질랜드 오클랜드 대학교의 로스, 카, 로버트 젠틀맨이 개발
 - ✓ 쉬운 편집/개발 환경
 - ✓ 데이터의 쉬운 조작
 - ✓ 풍부한 실습용 데이터 세트
 - ✓ 주변에 산재된 사회와 자연 현상의 데이터
 - ✓ 풍부한 라이브러리
 - ✓ 무료




프로그래밍 언어와 R

Oct 2020	Oct 2019	Change	Programming Language	Ratings	Change
1	2	⬆	C	16.95%	+0.77%
2	1	⬇	Java	12.56%	-4.32%
3	3		Python	11.28%	+2.19%
4	4		C++	6.94%	+0.71%
5	5		C#	4.16%	+0.30%
6	6		Visual Basic	3.97%	+0.23%
7	7		JavaScript	2.14%	+0.06%
8	9	⬆	PHP	2.09%	+0.18%
9	15	⬆	R	1.99%	+0.73%
10	8	⬇	SQL	1.57%	-0.37%
11	19	⬆	Perl	1.43%	+0.40%
12	11	⬇	Groovy	1.23%	-0.16%
13	13		Ruby	1.16%	-0.16%
14	17	⬆	Go	1.16%	+0.06%
15	20	⬆	MATLAB	1.12%	+0.19%



https://www.r-project.org/

통계 처리와 그래픽을 위한 무료 소프트웨어 환경



[Home]

Download

CRAN

The R Project for Statistical Computing

Korea

- <https://ftp.harukasa.co.kr/mirrors2/CRAN/>
- <https://cran.yu.ac.kr/>
- <https://cran.seoul.kr/>
- <http://healthstat.seoul.kr/cran/>
- <https://cran.bionics.ac.kr/>

Download and Install R

Precompiled binary distributions of the R graphics. It compiles and runs on a wide range of operating systems. For R, please choose your preferred CRAN

these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

Subdirectories:

[base](#)

Binaries for base distribution. This is what you want to **install R for the first time.**

[contrib](#)

Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on [third party software](#) available for CRAN Windows services and corresponding environment and make variables.

[old contrib](#)

Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).

[Rtools](#)

Tools to build R packages. This is needed if you want to build R packages on Windows, or to build R itself.

[Download R 4.0.3 for Windows](#)

85 megabytes, 32/64 bit)

[Installation and other instructions](#)

[New features in this version](#)




R studio 설치

R을 위한 통합 개발환경(IDE)

https://rstudio.com/

BLACK LIVES MATTER

JOIN US AND DONATE




Products ^

Solutions v


Customers

OPEN SOURCE

Get started with R

 RStudio

The premier IDE



RStudio Desktop

Run RStudio on your computer

Support

Community forums only

License

AGPL v3

Pricing

Free

DOWNLOAD RSTUDIO DESKTOP


RStudio Desktop

Open Source License

Free

DOWNLOAD

Learn more



DOWNLOAD RSTUDIO FOR WINDOWS

1.3.1093 | 171.62MB



R Package

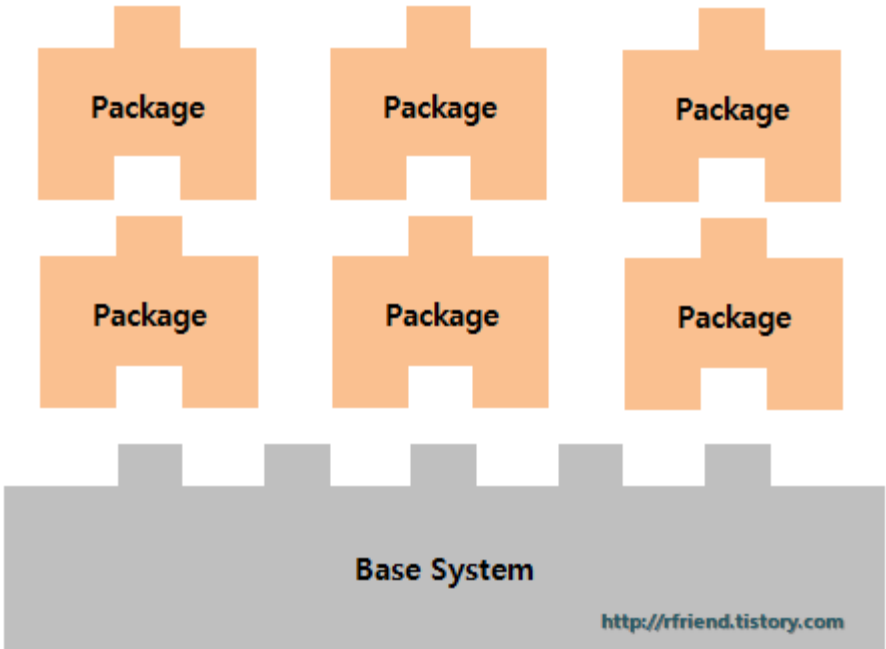
Package : 다양한 사용자들의 희망으로 생겨나는 많은 로직들과 코드들

【 설치와 사용방법에 따른 Package 구분 】

Package 구분	설치	사용
Base Packages	자동	불러오기 불필요
Recommended Packages	자동	불러오기 필요 library (package name)
Other Packages	개별 설치 install.packages ("package name")	불러오기 필요 library (package name)

<http://rfriend.tistory.com>

【 Base System과 Package의 관계 】



패키지(package)는 R 함수들을 모아 놓은 컬렉션이며, 라이브러리(library)는 R 패키지가 저장되는 폴더를 의미



R 시작하기

RGui (64-bit)

파일 편집 보기 기타 패키지를 윈도우2 도움말

복사하기 Ctrl+C

붙여넣기 Ctrl+V

명령어만 붙여넣기

복사 후 붙여넣기 Ctrl+X

모두 선택하기

콘솔 지우기 Ctrl+L

데이터 편집기...

GUI 설정...

R은 많은 기여자들이 참여하는 공동프로젝트입니다.

'contributors()'라고 입력하시면 이에 대한 더 많은 정보를 확인하

그리고, R 또는 R 패키지들을 출판물에 인용하는 방법에 대해서는 'cita

'demo()'를 입력하신다면 몇가지 데모를 보실 수 있으며, 'help()'

또한, 'help.start()'의 입력을 통하여 HTML 브라우저에 의한 도

R의 종료를 원하시면 'q()'를 입력해주세요.

[이전에 저장한 작업공간을 복구하였습니다]

> |

Rgui 구성 편집기

Single or multiple

☒ MDI ☐ SDI

☒ MDI toolbar ☐ MDI statusbar

Pager style

☒ multiple windows ☐ single window

Language for menus and messages

Font

d2coding

☒ TrueType only

size 14

style normal

Console rows 23 columns 77

☒ set options(width) on resize?

☒ buffer console by default?

Initial left 0 top 0

buffer chars 250000 lines 8000

Cursor blink Partial

Pager rows 25 columns 80

Graphics windows: initial left -25 top 0

Console and Pager Colours

background normaltext usertext pagerbg

wheat2 wheat3 wheat4 white

Sample text

Apply

Save...

Load...

OK

Cancel

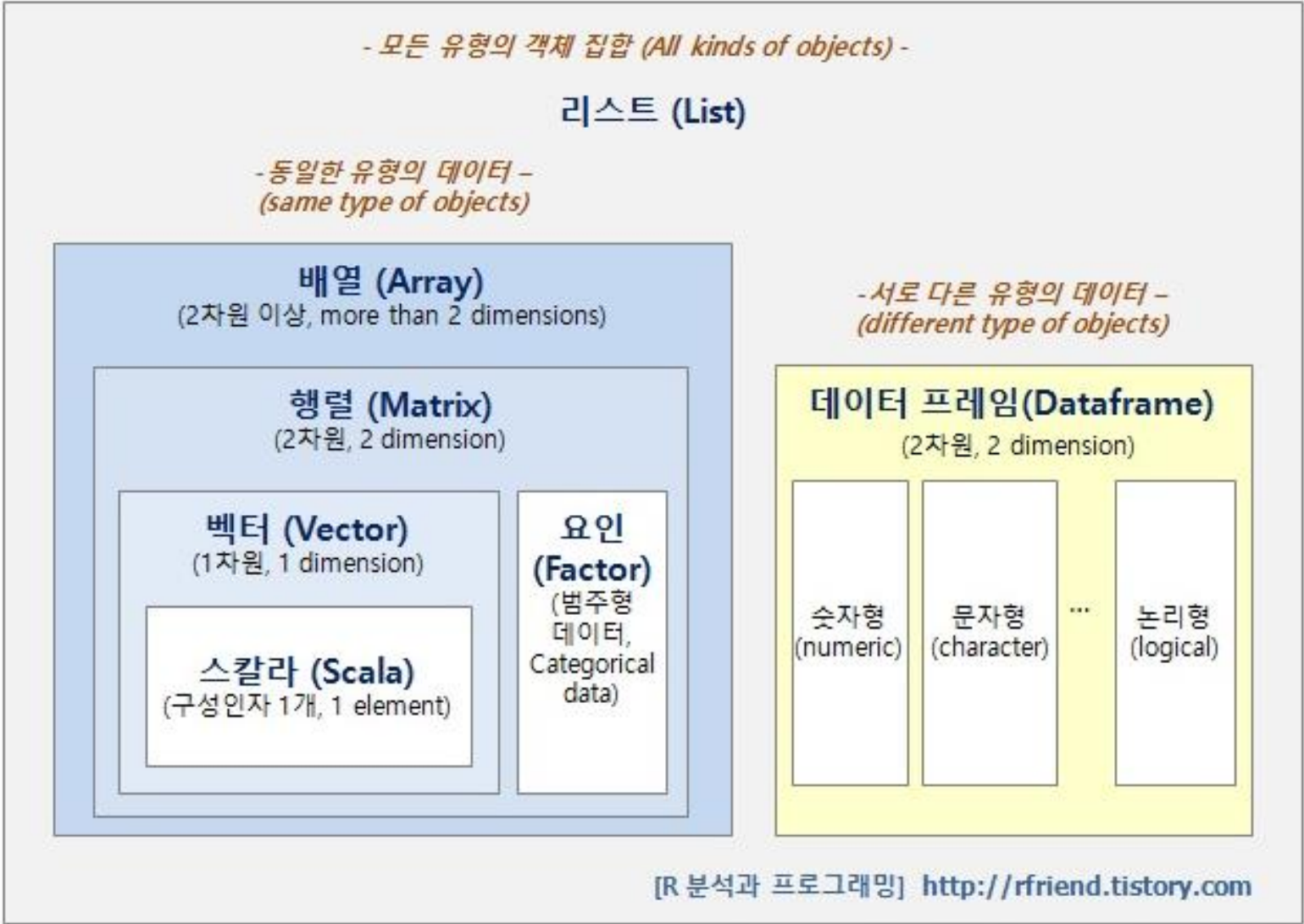
> x <- c(1, 2, 3, 4, 5)

> mean(x)

[1] 3



데이터 구조의 이해



Rule Variable in R

1. 변수 또는 함수를 다른 것과 구별하기 위해 사용하는 이름
2. 식별자 명명 규칙
 - ✓ 알파벳, 숫자, '_', '.'의 조합
 - ✓ 첫 글자는 알파벳이나 '.'으로 시작
 - ✓ R에서 사용하는 예약어(예: if, for, while, TRUE, NA 등)를 사용 불가

1. The variable name must start with letter and can contain number, letter, underscore(' ') and period('.').
2. Reserved words or keywords are not allowed to be defined as a variable name.
3. Special characters such as '#', '&', etc., along with White space (tabs, space) are not allowed in a variable name.

Variables in R can be assigned in one of three ways.

1. Assignment Operator: "=" used to assign the value.
ex) first.variable = 20
2. '<-' Operator
ex) second.variable <- "New Program"
3. '->' Operator
ex) 565 -> third.variable



1. 스칼라(Scala) : 구성 인자가 하나인 벡터

```
> #스칼라(Scala) : 구성인자가 1개인 벡터  
> n1 <- c(1)  
> s1 <- c("Kim")
```

2. 벡터(Vector) : 동일한 유형의 데이터가 구성 인자가 1개 이상이면서 1차원으로 구성된 데이터 구조

```
> #벡터(Vector)  
> v1 <- c(1, 2, 3)  
> v2 <- c("Kim", "Lee", "Choi")  
> v3 <- c(TRUE, FALSE, TRUE)
```

```
> v1 <- c(1, 2, 3, TRUE)  
> v1  
[1] 1 2 3 1  
> v2 <- c(1, 2, 3, "TRUE")  
> v2  
[1] "1"    "2"    "3"    "TRUE"
```

```
> help(c)  
starting httpd help server ... done  
> ?c
```



3. 요인(Factor) : 범주형(명목형 또는 순서형)의 데이터 구조

- ✓ 범주형 데이터 : 데이터가 사전에 정해진 특정 유형(목록)으로만 분류되는 경우를 뜻함.
- ✓ 사전에 정해진 특정 유형의 수를 레벨(Level)이라고 함.
- ✓ 명목형 데이터 : "남", "여"와 같이 값들 간에 크기 비교가 불가능한 경우를 뜻함.
- ✓ 순서형 데이터 : "대", "중", "소"와 같이 값에 순서를 둘 수 있는 경우를 뜻함.

```
> # (1) 문자형 데이터를 그냥 입력하면, 따옴표가 있는 문자형 벡터가 생성
```

```
> f1 <- c("Middle", "Low", "High")
```

```
> f1
```

```
[1] "Middle" "Low"      "High"
```

```
>
```

```
> # (2) factor() 함수를 이용해서 문자형 벡터를 요인(factor)로 변환
```

```
> # 단, 순서를 지정하지 않으면 알파벳 순서로 수준(level)이 자동으로 지정됨
```

```
> f2 <- factor(f1)
```

```
> f2
```

```
[1] Middle Low      High
```

```
Levels: High Low Middle
```

```
>
```

```
> # (3) 수준(level)에서 순서를 부여하려면 'order=TRUE' 옵션 설정, level=c("")에 순서대로 입력
```

```
> f3 <- factor(f2, order=TRUE, level=c("Low", "Middle", "High"))
```

```
> f3
```

```
[1] Middle Low      High
```

```
Levels: Low < Middle < High
```




```
> #yes, no를 명목형 범주로 지정하여 해당 값만 넣을 수 있다.  
> answer <- factor("yes", c("yes","no"))  
> answer  
[1] yes  
Levels: yes no  
> #범주값을 벗어나는 값은 넣을 수 없다.  
> answer <- factor(c("why","yes","good"), c("yes","no"))  
> answer  
[1] <NA> yes <NA>  
Levels: yes no  
> # 순서형 범주로 레벨 순서를 줄 수 있다.  
> order1 <- factor(1,c(1,2,3),ordered = TRUE,levels = c(3,2,1))  
> order1  
[1] 3  
Levels: 1 < 2 < 3
```



4. 행렬(Matrix) : 동일한 유형의 2차원 데이터 구조(행(Row), 열(Column)) 벡터는 동일한 유형의 1차원 데이터 구조

```
> #1-12까지의 숫자를 행(row)의 수가 4개인 행렬로 만들어라.  
> m1 <- matrix(1:12, nrow=4)  
> m1  
      [,1] [,2] [,3]  
[1,]    1    5    9  
[2,]    2    6   10  
[3,]    3    7   11  
[4,]    4    8   12  
>  
> #1-12까지의 숫자를 행(row)의 수가 4개인 행렬을 만드는데, 행기준으로 만들어라.  
> m2 <- matrix(1:12, nrow=4, byrow=TRUE)  
> m2  
      [,1] [,2] [,3]  
[1,]    1    2    3  
[2,]    4    5    6  
[3,]    7    8    9  
[4,]   10   11   12
```



```
> y <- matrix(c(1,2,3,4),nrow=2,byrow=T) #행렬만들기
> y
```

```
      [,1] [,2]
[1,]    1    2
[2,]    3    4
```

```
> y %*% y #행렬간 곱하기
```

```
      [,1] [,2]
[1,]    7   10
[2,]   15   22
```

```
> 3*y      #모든 원소에 *3
```

```
      [,1] [,2]
[1,]     3     6
[2,]     9    12
```

```
> y*y      #각 원소의 제곱
```

```
      [,1] [,2]
[1,]     1     4
[2,]     9    16
```

```
> z<-matrix(c(1,1,1,2,1,0,3,0,1,4,0,0),nrow=4,byrow=T) #행렬 만들기
> z
```

```
      [,1] [,2] [,3]
[1,]     1     1     1
[2,]     2     1     0
[3,]     3     0     1
[4,]     4     0     0
```

```
> z %*% z      #에러 (4*3) * (4*3)
Error in z %*% z : 적합한 인자들이 아닙니다
```

```
> z %*% t(z) #전치행렬 ((4*3) * (3*4))
```

```
      [,1] [,2] [,3] [,4]
[1,]     3     3     4     4
[2,]     3     5     6     8
[3,]     4     6    10    12
[4,]     4     8    12    16
```

```
> z[,2:3]      #행은 모두 출력하고 2, 3번째 열만 출력
```

```
      [,1] [,2]
[1,]     1     1
[2,]     1     0
[3,]     0     1
[4,]     0     0
```

```
> z[-2]      #2행을 지우고 모두 출력
```

```
[1] 1 3 4 1 1 0 0 1 0 1 0
```



```
> dim(z)    #행렬이 몇행 몇열인지 출력  
[1] 4 3  
> dim(z)[1] # z행렬의 행출력  
[1] 4  
> dim(z)[2] # z행렬의 열출력  
[1] 3
```

```
> z <- matrix(c(1,2,3,4,5,6),nrow=3,byrow=F)    #행열 만들기  
> z  
      [,1] [,2]  
[1,]    1    4  
[2,]    2    5  
[3,]    3    6  
> apply(z,1,mean)    #(1,4)의 평균, (2,5)의 평균, (3,6)의 평균  
[1] 2.5 3.5 4.5  
> apply(z,2,mean)    #(1,2,3,)의 평균, (4,5,6)의 평균  
[1] 2 5  
> apply(z,1,sum)     #(1,4)의 합, (2,5)의 합, (3,6)의 합  
[1] 5 7 9  
> apply(z,2,sum)     #(1,2,3)의 합, (4, 5, 6)의 합  
[1] 6 15
```



```
> a1 <- array(1:24, c(2,3,4))
> a1
, , 1
      [,1] [,2] [,3]
[1,]     1     3     5
[2,]     2     4     6

, , 2
      [,1] [,2] [,3]
[1,]     7     9    11
[2,]     8    10    12

, , 3
      [,1] [,2] [,3]
[1,]    13    15    17
[2,]    14    16    18

, , 4
      [,1] [,2] [,3]
[1,]    19    21    23
[2,]    20    22    24
```

5. 배열(Array) : 한 개 이상의 벡터로 구성, 동일한 데이터 유형

```
> arr1 <- array(1:12, dim=c(2,2,3))
> arr1
, , 1
      [,1] [,2]
[1,]     1     3
[2,]     2     4

, , 2
      [,1] [,2]
[1,]     5     7
[2,]     6     8

, , 3
      [,1] [,2]
[1,]     9    11
[2,]    10    12
```



6. 데이터 프레임(Data frame) : 데이터 유형에 상관없이 2차원 형태의 데이터 구조

```
> # 다른 유형의 벡터 생성
> d1 <- c(1,2,3,4)
> d2 <- c("Kim", "Lee", "Choi", "Park")
>
> # 데이터 프레임으로 묶기 : data.frame() 함수 사용
> d3 <- data.frame(cust_id = d1, last_name = d2) # 필드명 부여
> d3
  cust_id last_name
1       1      Kim
2       2      Lee
3       3     Choi
4       4     Park
```

```
> x <- data.frame(성명=c("홍길동", "손오공"), 나이=c(20, 30), 주소=c("서울", "부산"))
> x
  성명 나이 주소
1 홍길동  20 서울
2 손오공  30 부산
```




```
> x <- cbind(x, 학과=c("전산학", "경영학"))
> x
```

	성명	나이	주소	학과
1	홍길동	20	서울	전산학
2	손오공	30	부산	경영학

```
> x <- rbind(x, data.frame(성명="장발장", 나이=40, 주소="파리", 학과="전산학"))
> x
```

	성명	나이	주소	학과
1	홍길동	20	서울	전산학
2	손오공	30	부산	경영학
3	장발장	40	파리	전산학

```
> name <- c("팬텀", "크리스틴 다예", "라울드 샤니")
> gender <- c("남", "여", "남")
> score.eng <- c(80, 90, 100)
> score.phil <- c(90, 95, 87)
> df <- data.frame(성명=name, 성별=gender, 영어=score.eng, 철학=score.phil)
> df
```

	성명	성별	영어	철학
1	팬텀	남	80	90
2	크리스틴 다예	여	90	95
3	라울드 샤니	남	100	87



7. 리스트(List) : 다양한 형태의 데이터 유형을 갖는 각 개체를 표현하는데 사용

```
> list1 <- list(x=1:4, y=c("가", "나"))
> list1
$x
[1] 1 2 3 4

$y
[1] "가" "나"

> list1$x
[1] 1 2 3 4
> list1[1]
$x
[1] 1 2 3 4
```

```
> list2 <- list(name="팬텀", gender="남", score=c(85,90))
> list2
$name
[1] "팬텀"

$gender
[1] "남"

$score
[1] 85 90

> list2$score
[1] 85 90
>
> list2$score[1]
[1] 85
```



```
> # Vector(L1), Matrix(L2), Array(L3), Data Frame(L4)를 만들어서, 하나의 List(L5)로 묶어라
> L1 <- c(1, 2, 3, 4) # Vector
> L2 <- matrix(1:6, 3, byrow=TRUE) # Matrix
> L3 <- array(1:24, c(3,4,2)) # Array
> L4 <- data.frame(cust_id = c(1, 2, 3, 4), last_name = c("Kim", "Lee", "Choi", "Park")) # Data Frame
> L5 <- list(L1, L2, L3, L4) # List
> L5
```

```
[[1]]
[1] 1 2 3 4
```

```
[[2]]
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

```
[[3]]
, , 1
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

```
, , 2
      [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

```
[[4]]
  cust_id last_name
1        1      Kim
2        2      Lee
3        3      Choi
4        4      Park
```



R에서 제공하는 데이터 세트 불러오기

```
> data()
```

Data sets in package 'datasets':

AirPassengers	Monthly Airline Passenger Numbers 1949-1960
BJsales	Sales Data with Leading Indicator
BJsales.lead (BJsales)	Sales Data with Leading Indicator
BOD	Biochemical Oxygen Demand
CO2	Carbon Dioxide Uptake in Grass Plant
ChickWeight	Weight versus age of chicks on diffe
DNase	Elisa assay of DNase
EuStockMarkets	Daily Closing Prices of Major Europe Indices, 1991-1998
Formaldehyde	Determination of Formaldehyde
HairEyeColor	Hair and Eye Color of Statistics Stu
Harman23.cor	Harman Example 2.3
Harman74.cor	Harman Example 7.4
Indometh	Pharmacokinetics of Indomethacin
InsectSprays	Effectiveness of Insect Sprays
JohnsonJohnson	Quarterly Earnings per Johnson & Joh
LakeHuron	Level of Lake Huron 1875-1972
LifeCycleSavings	Intercountry Life-Cycle Savings Data
Loblolly	Growth of Loblolly pine trees
Nile	Flow of the River Nile
Orange	Growth of Orange Trees
OrchardSprays	Potency of Orchard Sprays

```
> quakes
```

	lat	long	depth	mag	stations
1	-20.42	181.62	562	4.8	41
2	-20.62	181.03	650	4.2	15
3	-26.00	184.10	42	5.4	43
4	-17.97	181.66	626	4.1	19
5	-20.42	181.96	649	4.0	11
6	-19.68	184.31	195	4.0	12
7	-11.70	166.10	82	4.8	43
8	-28.11	181.93	194	4.4	15
9	-28.74	181.74	211	4.7	35
10	-17.47	179.59	622	4.3	19



Software platform

```
> head(quakes, n=10) #데이터의 앞부분 10개(디폴트=6)
      lat  long depth mag stations
1 -20.42 181.62  562 4.8       41
2 -20.62 181.03  650 4.2       15

> tail(quakes, n=6) #데이터의 뒷부분 일부 보기(디폴트=6)
      lat  long depth mag stations
995 -17.70 188.10   45 4.2        10
996 -25.93 179.54  470 4.4        22
997 -12.28 167.06  248 4.7        35
998 -20.13 184.20  244 4.5        34
999 -17.40 187.80   40 4.5        14
1000 -21.59 170.56  165 6.0       119

> names(quakes) #'quakes' 데이터 세트의 변수명 보기
[1] "lat"      "long"     "depth"    "mag"      "stations"
```

```
> str(quakes) #'quakes' 데이터 세트의 차원 보기(행과 열의 수)
'data.frame':  1000 obs. of  5 variables:
 $ lat      : num  -20.4 -20.6 -26 -18 -20.4 ...
 $ long     : num   182 181 184 182 182 ...
 $ depth    : int   562 650 42 626 649 195 82 194 211 622 ...
 $ mag      : num   4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
 $ stations: int   41 15 43 19 11 12 43 15 35 19 ...
```

```
> summary(quakes) #'quakes' 데이터 세트의 각 변수별 데이터 요약 보기
      lat      long      depth      mag
Min.   :-38.59  Min.   :165.7  Min.    : 40.0  Min.    :4.00
1st Qu.: -23.47  1st Qu.:179.6  1st Qu.: 99.0  1st Qu.:4.30
Median : -20.30  Median :181.4  Median :247.0  Median :4.60
Mean    : -20.64  Mean    :179.5  Mean    :311.4  Mean    :4.62
3rd Qu.: -17.64  3rd Qu.:183.2  3rd Qu.:543.0  3rd Qu.:4.90
Max.    : -10.72  Max.    :188.1  Max.    :680.0  Max.    :6.40
 stations
Min.    : 10.00
1st Qu.: 18.00
Median  : 27.00
Mean    : 33.42
3rd Qu.: 42.00
Max.    :132.00
```

```
> summary(quakes$mag) #'quakes' 데이터 세트 내의 msg 변수 정보 보기
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      4.00   4.30   4.60   4.62   4.90   6.40
```

```
> write.table(quakes, "c:/temp/quakes.txt", sep=",")
```

외부 파일 불러오기

```
> setwd("c:/datamarket") #작업 디렉토리 지정
> getwd()                #작업 디렉토리 확인
[1] "c:/datamarket"
> exam_df<-read.csv("exam.csv",header=TRUE) #파일 읽어오기
> head(exam_df)          #읽어온 데이터 확인하기
  exam1 exam2 quiz
1      3      2   3
2      5      1   5
3      4      5   5
4      2      6   3
5      5      6   7
6      2      5   7
> str(exam_df)           #데이터의 수와 구조 확인하기
'data.frame':  11 obs. of  3 variables:
 $ exam1: int  3 5 4 2 5 2 4 8 9 4 ...
 $ exam2: int  2 1 5 6 6 5 5 5 4 7 ...
 $ quiz : int  3 5 5 3 7 7 3 6 4 6 ...
```

```
> x <- read.csv(file.choose(), header=T)
> x
```

	A	B	C
1	exam1	exam2	quiz
2	3	2	3
3	5	1	5
4	4	5	5
5	2	6	3
6	5	6	7
7	2	5	7
8	4	5	3
9	8	5	6
10	9	4	4
11	4	7	6
12	5	9	2
13			



	A	B	C
1	exam1	exam2	quiz
2	3	2	3
3	5	1	5
4	4	5	5
5	2	6	3
6	5	6	7
7	2	5	7
8	4	5	3
9	8	5	6
10	9	4	4
11	4	7	6
12	5	9	2
13			

데이터 프레임의 index 접근 방법

1. matrix와 동일한 접근법 #exam_df[2:3,1] or exam_df[2:3,"exam1"]
2. column명 이용 #exam_df\$exam1[2:3]
3. TRUE, FALSE를 이용 #이후설명

```
> exam_df$exam1[2:3]
[1] 5 4
> exam_df$exam1[c(2,3)]
[1] 5 4
> exam_df[2:3,1]
[1] 5 4
```

```
> index_vector<-rep(FALSE,11)
> index_vector[2:3]<-TRUE
> index_vector
[1] FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
>
> exam_df[index_vector,1]
[1] 5 4
```

```
> index_vector<-exam_df$exam1<3
> index_vector
[1] FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE
> exam_df[index_vector,1]
[1] 2 2
```



```
> index_wh<-which(exam_df$exam1<3)
> index_wh
[1] 4 6
> exam_df[index_wh,1]
[1] 2 2
```

```
> index<-rep(FALSE,11*3)
> index[13:14]<-TRUE
> index_matrix<-matrix(index,ncol=3)
> index_matrix
      [,1] [,2] [,3]
[1,] FALSE FALSE FALSE
[2,] FALSE TRUE  FALSE
[3,] FALSE TRUE  FALSE
[4,] FALSE FALSE FALSE
[5,] FALSE FALSE FALSE
[6,] FALSE FALSE FALSE
[7,] FALSE FALSE FALSE
[8,] FALSE FALSE FALSE
[9,] FALSE FALSE FALSE
[10,] FALSE FALSE FALSE
[11,] FALSE FALSE FALSE
> exam_df[index_matrix]
[1] 1 5
> which(index_matrix)
[1] 13 14
```



웹 파일 불러오기

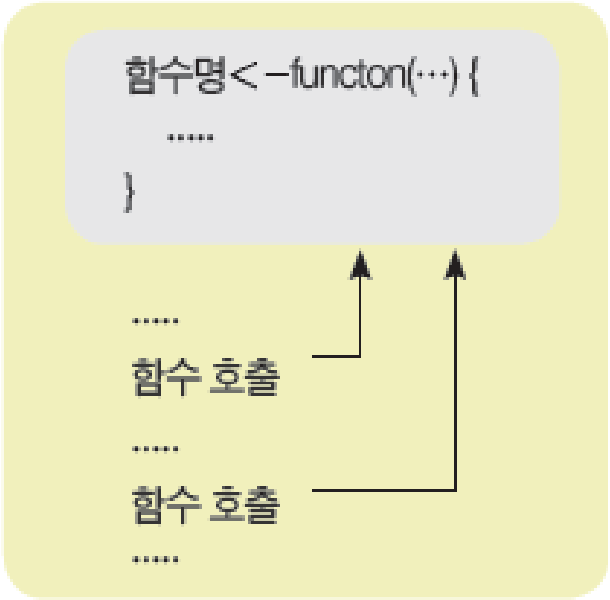
```
> url <- "https://vincentarelbundock.github.io/Rdatasets/csv/datasets/Titanic.csv"
> x <- read.csv(url)
> x
```

	X	Class	Sex	Age	Survived	Freq
1	1	1st	Male	Child	No	0
2	2	2nd	Male	Child	No	0
3	3	3rd	Male	Child	No	35
4	4	Crew	Male	Child	No	0
5	5	1st	Female	Child	No	0
6	6	2nd	Female	Child	No	0
7	7	3rd	Female	Child	No	17
8	8	Crew	Female	Child	No	0
9	9	1st	Male	Adult	No	118
10	10	2nd	Male	Adult	No	154



함수 만들기

■ 함수: 하나 이상의 명령어로 반복 사용 가능한 기능을 구현한 것



함수명

반지름의 길이를 매개변수(parameter)로 함
⇨ 매개변수가 없거나, 1개 이상도 가능함. 1개 이상의 경우, 쉼표(,)로 구분함

함수의 정의

```
getCircleArea <- function(r)  
  area = 3.14 * r^2  
  return(area)  
}
```

반환 값(반환 값이 없을 경우, 생략)

함수의 호출

```
getCircleArea(3)
```

인자 또는 인수(argument)
⇨ 정의된 함수의 매개변수 수와 같게 함

출력 결과

```
[1] 28.26
```



reference

빅데이터분석의 첫걸음 R로 배우는 코딩, 장용식, 강희구, 생능출판
머신러닝을 활용한 R 데이터 분석, 장용식, 최진호, 생능출판

<https://rc2e.com/>

http://www.datamarket.kr/xe/board_ecko11

<https://www.statmethods.net/index.html>

<http://www.tagxedo.com/>

<https://magic.piktochart.com/>

<https://infogr.am/>

