

4. 프로그래밍(R)

강의자 : 조상흠 박사님



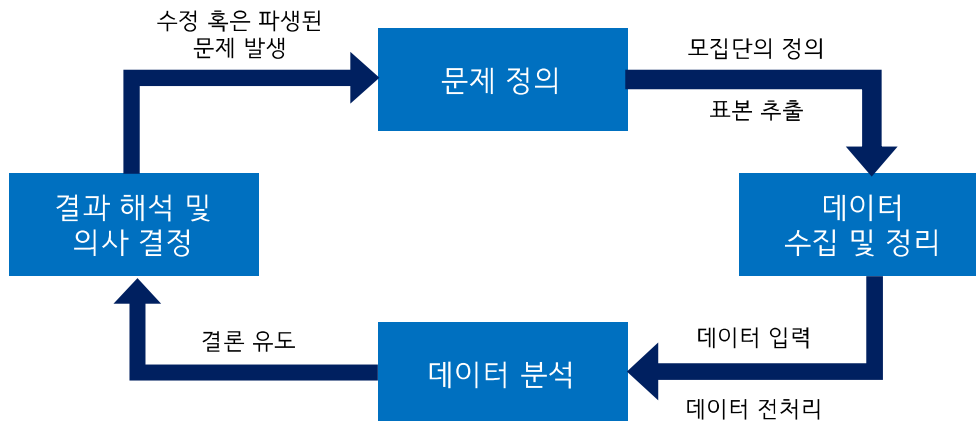
R 프로그래밍을 활용한 통계분석

DFMBA 사전교육 - R프로그래밍
조상흠

R프로그래밍과 통계분석

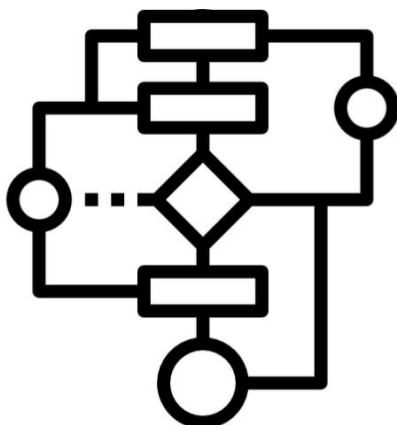
통계 분석 도구로서의 R 프로그래밍

- R 프로그램은 통계 분석을 위한 언어이자 소프트웨어 도구다.
- 통계 분석이란? 특정 집단을 대상으로 자료를 수집하여 대상 집단에 대한 정보를 구하고, 적절한 통계분석 방법론을 이용하여 의사결정 (통계적 추론)을 하는 과정



프로그래밍이란?

- 알고리즘 (Algorithm)
 - 문제 해결에 필요한 기본적 연산들을 명확하고 정확한 순서로 나열한 것.



- Ex) Call option 가치 구하기
 - Strike Price (행사가격) : K
 - Future stock price : S_T
 - Calculate $S_T - K$
 - Calculate $\max(S_T - K, 0)$
 - 사용된 연산 : $-, \max()$

프로그래밍이란?

- 프로그래밍 언어 : 해당 알고리즘을 컴퓨터로 구현.
- 데이터 처리, 연산, 통계적 분석, 자료 저장 등



R 프로그래밍

- 통계 분석과 그래픽을 제공하는 무료 프로그램! Free Software
- 다양한 운영체제(Windows, MacOS, Linux) 에서 사용 가능.
- 매우 폭넓은 사용자층.
- 풍부한 프로그램 소스. Open source.
- 직관적인 코드.
- 접근성과 범용성이 편리하며,
- 수많은 통계 프로젝트를 쉽게 수행 가능!
- 금융, 기계학습, 데이터마이닝, 경제학, 심리학



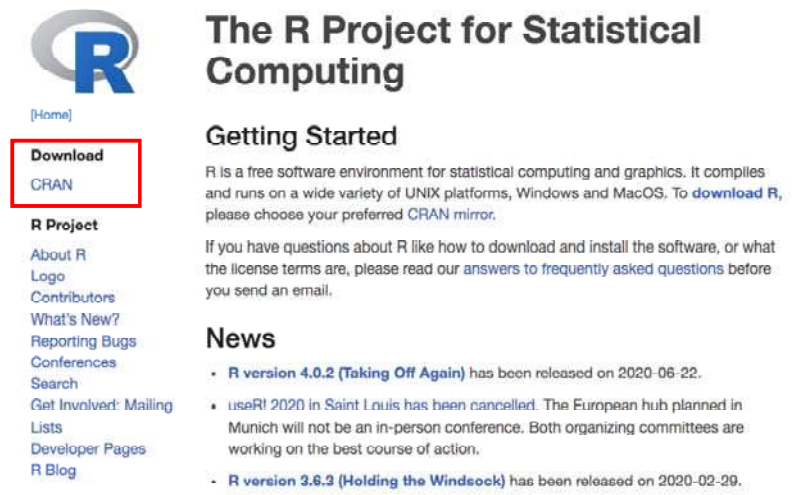
R 프로그래밍: 유용한 사이트들

- R Foundation : <http://www.r-project.org>
 - R에 관한 정보
- CRAN (The Comprehensive R Archive Network)
: <http://cran.r-project.org>
 - 패키지(Packages)들 및 자료 다운로드
- Bioconductor : www.bioconductor.org
- GitHub : github.com
- Google, google, google, google, ... (코딩은 구글링의 연속이다!!)

R프로그램 설치

R 설치

- <http://www.r-project.org>



The R Project for Statistical Computing

[Home]

Download
CRAN

R Project
About R
Logo
Contributors
What's New?
Reporting Bugs
Conferences
Search
Get Involved: Mailing Lists
Developer Pages
R Blog

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

News

- **R version 4.0.2 (Taking Off Again)** has been released on 2020-06-22.
- **useR! 2020 in Saint Louis has been cancelled.** The European hub planned in Munich will not be an in-person conference. Both organizing committees are working on the best course of action.
- **R version 3.6.3 (Holding the Windsock)** has been released on 2020-02-29.

R 설치

- CRAN Mirror 고르기

| | | |
|-----------|---|---|
| Indonesia | https://repo.bnpf.go.id/cran/ | Agency for The Application and Assessment of Technology |
| Iran | https://cran.um.ac.ir/ | Ferdowsi University of Mashhad |
| Ireland | https://ftp.heanet.ie/mirrors/cran.r-project.org/ | HEAnet.Dublin |
| Italy | http://cran.mirror.garr.it/mirrors/CRAN/ https://cran.stat.unipd.it/ | Garr Mirror, Milano University of Padua |
| Japan | https://cran.ism.ac.jp/ https://ftp.yz.yamagata-u.ac.jp/pub/cran/ | The Institute of Statistical Mathematics, Tokyo Yamagata University |
| Korea | https://cran.yu.ac.kr/ http://healthstat.snu.ac.kr/C-RAN/ https://cran.bioidisk.org/ | Yeungnam University Graduate School of Public Health, Seoul National University, Seoul The Genome Institute of UNIST (Ulsan National Institute of Science and Technology) |
| Malaysia | https://wbc.upm.edu.my/cran/ | Universiti Putra Malaysia |
| Mexico | https://cran.itam.mx/ http://www.est.colpos.mx/R-mirror/ | Instituto Tecnológico Autónomo de México Colegio de Postgraduados, Texcoco |
| Morocco | https://mirror.marwan.ma/cran/ | MARWAN |

R 설치

• 운영체제 선택하여 설치

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#) (Debian, Fedora/Redhat, Ubuntu)
- [Download R for macOS](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2021-08-10, Kick Things) [R-4.1.1.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

R 설치

• Windows



CRAN

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

[About R](#)

[R Homepage](#)

[The R Journal](#)

[Software](#)

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

R-4.0.2 for Windows (32/64 bit)

[Download R 4.0.2 for Windows](#) (84 megabytes, 32/64 bit)

[Installation and other instructions](#)

[New features in this version](#)

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the file to the [checksum](#) on the master server. You will need a version of md5sum for windows: both [gchical](#) and [command line versions](#) are available.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Other builds

[R64bit for Windows 64-bit](#)

• MacOS

R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) [here](#). Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the [p10](#) directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

Package binaries for R versions older than 3.2.0 are only available from the [CRAN archive](#) so users of such versions should adjust the CRAN mirror setting ([https://cran.archive.r-project.org](#)) accordingly.

R 4.0.2 "Taking Off Again" released on 2020/06/22

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
md5 R-4.0.2.pkg
in the *Terminal* application to print the MD5 checksum for the R-4.0.2.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
`shasum -l -csha256-signature R-4.0.2.pkg`

Latest release:

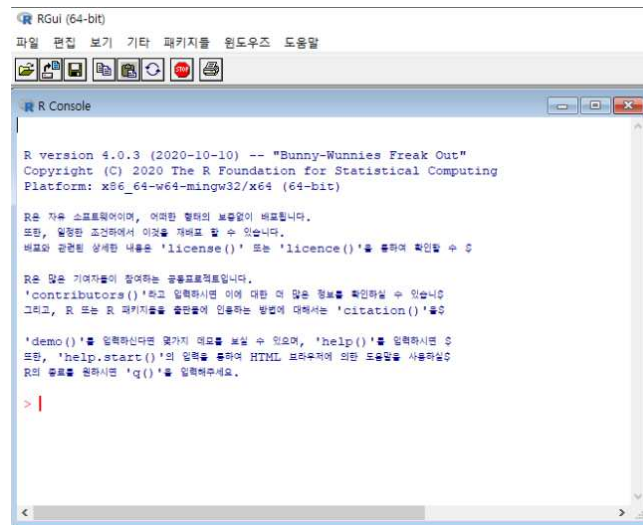
R-4.0.2.pkg (notarized and signed)

md5: 7b6b7f04d27c2d475e6a6d0b6a20b6b83d3b3
(via: SHA256)

R 4.0.2 binary for macOS 10.13 (High Sierra) and higher, signed and notarized package. Contains R 4.0.0 framework, Rapp GUI 1.72 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texpdf 6.7. The latter two components are optional and can be omitted when choosing "custom install", they are only needed if you want to use the `scx1x` R package or build package documentation from sources.

R 설치

- R 실행화면



```
R GUI (64-bit)
파일 편집 보기 기타 패키지를 윈도우즈 도움말

R Console

R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R은 자유 소프트웨어이며, 어떠한 형태의 보증없이 배포됩니다.
또한, 일정한 조건하에서 이것을 재배포 할 수 있습니다.
배포와 관련된 상세한 내용은 'license()' 또는 'licence()'를 통하여 확인할 수 있습니다.


R은 많은 기여자들이 참여하는 공동프로젝트입니다.
'contributors()'라고 입력하시면 이에 대한 더 많은 정보를 확인할 수 있습니다.
그리고, R 또는 R 패키지를 출판물에 인용하는 방법에 대해서는 'citation()'를 참조하십시오.

'demo()'를 입력하신다면 몇가지 예제를 보실 수 있으며, 'help()'를 입력하시면 R
또한, 'help.start()'의 입력을 통하여 HTML 브라우저에 의한 도움말을 사용할 수 있습니다.
R의 도움을 원하시면 'q()'를 입력하십시오.

> |
```

R 코딩 기초

R 코딩 기초

- New Document : (Source) Code를 작성 및 저장하는 작업공간 생성
 - R 실행 → [File] → [New script]
 - —  Source Code 불러오기 : [File] → [Open script]

R 코딩 기초

[실습1]

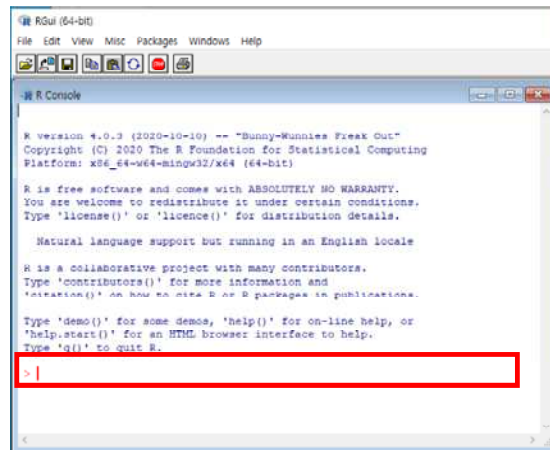
- 이 강의의 Source Code가 담긴 파일
‘R프로그래밍을 활용한 통계분석 lecture code’ document를 불러옵니다.



```
## Example2
3
1+1
5-7
pi
sin(pi/4)
sqrt(2)
1/sqrt(2)
```

R 코딩 기초

- > : Console 창에서 입력을 기다리는 prompt.
여기에 명령(코드; Code)를 입력한다.



R 코딩 기초

- > : 입력을 기다리는 prompt. 여기에 명령(코드; Code)를 입력한다.
 - 한 줄씩 직접 입력하여 코딩.
 - Document에서 필요한 부분을 복사하여 코딩.

```
C:\Google Drive\Chun\KAIST\TA\FM8

## Example2
3
1+1
5-7
pi
sin(pi/4)
sqrt(2)
1/sqrt(2)
```



```
> 3
[1] 3
> 1+1
[1] 2
> |
```

R 코딩 기초

[실습2]

- 이 강의의 Source Code Document에서, 'Exercise 2' 부분을 Console창에 직접 코딩하여 실행해 봅시다.

```
> 3
[1] 3
> 1+1
[1] 2
> 5-7
[1] -2
> pi
[1] 3.141593
> sin(pi/4)
[1] 0.7071068
> sqrt(2)
[1] 1.414214
> 1/sqrt(2)
[1] 0.7071068
> |
```

R 코딩 기초

계산 연산자

| Operator | 뜻 | 예시 |
|----------|------------|-------------------|
| + | 덧셈 | $8 + 8 = 16$ |
| - | 뺄셈 | $7 - 9 = -2$ |
| * | 곱셈 | $16 * 4 = 64$ |
| / | 나눗셈 | $5 / 2 = 2.5$ |
| %%/% | (정수) 나눗셈 몫 | $5 \% / \% 2 = 2$ |
| %% | 나머지 | $9 \% \% 3 = 0$ |
| ** | 거듭제곱 | $7 ** 2 = 49$ |

R 코딩 기초

기본 수학 함수

| Functions | 뜻 | 예시 |
|-----------|--------------|------------------|
| sqrt(x) | 루트 | sqrt(5)=2.236068 |
| exp(x) | 지수함수 | exp(1)=2.718282 |
| log(x) | 로그함수 | log(2)=0.6931472 |
| log10(x) | 상용로그함수 | log10(2)=0.30103 |
| sin(x) | sin함수 | sin(pi/2) = 1 |
| abs(x) | 절댓값 | abs(-32)=32 |
| floor(x) | x보다 작은 최대 정수 | floor(-8.2)=-9 |

R 코딩 기초

Comment (주석)

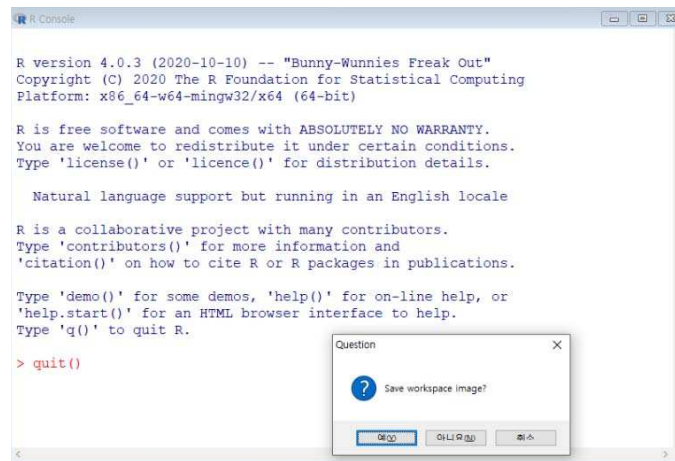
- 주석 : 코드 이해를 돕기 위해 덧붙이는 설명.
 - 어려운 코드에 대한 보충 설명
 - 추후 작업을 위한 체크리스트 역할.
- 코드 흐름에 영향을 주지 않음.
- 반드시 # 이후에 입력. #를 붙이지 않으면 Error 발생

```
> log(20) # natural log with base 'e'
[1] 2.995732
> log(20) natural log with base 'e'
Error: unexpected symbol in "log(20) natural"
```

R 코딩 기초

Quit R

- Console에 quit() 입력.
- Console 저장 시 [예], 그렇지 않으면 [아니요] 선택.



데이터 종류

데이터 종류

| 자료 형태 | 구성 차원 | 자료 유형 | 복수 데이터 유형 적용 여부 |
|-------------------|--------|-------------------------------|-----------------|
| 벡터(Vector) | 1차원 | 수치/문자/복소수/논리 | 불가능 |
| 행렬(Matrix) | 2차원 | 수치/문자/복소수/논리 | 불가능 |
| 데이터프레임(Dataframe) | 2차원 | 수치/문자/복소수/논리 | 가능 |
| 배열(Array) | 2차원 이상 | 수치/문자/복소수/논리 | 불가능 |
| 요인(Factor) | 1차원 | 수치/문자 | 불가능 |
| 시계열(Time Series) | 2차원 | 수치/문자/복소수/논리 | 불가능 |
| 리스트(list) | 2차원 이상 | 수치/문자/복소수/논리 함수/표현식/Call 등 | 가능 |

데이터 종류: 벡터 (Vector)

- 벡터 (Vector) : 동일한 형태의 자료를 1차원의 형태로 여러 개 모아서 취급하는 데이터 형태

```
> x <- c(1,2,3,4,5) # 벡터형 자료를 생성해서 변수 x에 할당한다
> x
[1] 1 2 3 4 5
>
> y <- rnorm(30) # 30개의 정규분포를 따르는 난수를 생성해서 변수 y에 할당한다
> y
[1] -1.13330968  2.15671166  0.90602539 -0.26011144  0.54052691 -0.08045706
[7]  0.44199533 -1.66020836 -0.11918926  0.58237450 -1.12243126  0.89946276
[13]  0.03454754  0.39217698 -1.13722361  1.07373194 -0.96200286  0.06244340
[19] -0.61888939 -1.23530584  1.07232747 -0.91446663  0.41121156  1.37899990
[25]  0.09922578  1.22303420  0.13543843 -0.53257020  0.19924238 -1.52453987
```

데이터 종류: 벡터 (Vector)

벡터 연산

```
> length(x)
[1] 5
> mean(x)
[1] 3
> sd(x)
[1] 1.581139
> range(x)
[1] 1 5
> length(x)
[1] 5
> sort(x)
[1] 1 2 3 4 5
> sort(x, decreasing = TRUE)
[1] 5 4 3 2 1
```

데이터 종류: 벡터 (Vector)

벡터 원소 다루기

```
> x <- c(1,3,5,7,9)
> x
[1] 1 3 5 7 9
> x[2] # x의 두 번째 원소
[1] 3
> x[-2] # x의 두 번째 원소를 제외한 나머지 원소들
[1] 1 5 7 9
> x[3]<-4 # x의 세번째 원소(5)를 4로 바꿈
> x
[1] 1 3 4 7 9
> x[2<x&x<5] # 2<x<5인 원소들 모두 출력
[1] 3 4
> y<-replace(x, c(2,4), c(11,13)) # x의 2,4번째 원소를 11, 13으로 바꿈
> y
[1] 1 11 4 13 9
```


데이터 종류: 행렬 (Matrix)

- 행렬(Matrix) : 동일한 형태의 자료를 2차원 형태로 여러 개 모아서 취급하는 데이터 형태
 - 가장 일반적으로 사용되는 자료형.
 - 숫자와 문자가 섞여서 구성될 수 없음.
- 행렬 생성 방법
 - 길이가 같은 벡터들을 만들고, rbind 혹은 cbind 이용
 - 함수 'matrix'와 parameter 'nrow' 이용

데이터 종류: 행렬 (Matrix)

행렬(Matrix) 생성

- 길이가 같은 벡터들을 만들고, rbind 혹은 cbind 이용

```
> vec1 <- c(1,2,3) # 벡터 (1,2,3) 생성
> vec2 <- c(4,5,6) # 벡터 (4,5,6) 생성
> vec3 <- c(7,8,9) # 벡터 (7,8,9) 생성
> mat1 <- rbind(vec1, vec2, vec3) # row (행) 방향으로 행렬 생성
> mat1
  [,1] [,2] [,3]
vec1  1   2   3
vec2  4   5   6
vec3  7   8   9
> mat2 <- cbind(vec1, vec2, vec3) # column (열) 방향으로 행렬 생성
> mat2
  vec1 vec2 vec3
[1,]  1   4   7
[2,]  2   5   8
[3,]  3   6   9
```

rbind(row + bind) : 행들을 이어 붙여서 행렬 생성

cbind(column + bind) : 열들을 이어 붙여서 행렬 생성

데이터 종류: 행렬 (Matrix)

행렬(Matrix) 생성

- 함수 'matrix'와 parameter 'nrow' 이용

```
> mat3 <- matrix(1:12, nrow = 3) # 3개의 row를 가진 3*4 행렬 생성
> mat3
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> mat4 <- matrix(1:12, nrow = 3, dimnames = list(c("R1", "R2", "R3"),
+ c("C1", "C2", "C3", "C4"))) # 행과 열 이름 설정
> mat4
      C1 C2 C3 C4
R1    1  4  7 10
R2    2  5  8 11
R3    3  6  9 12
```

데이터 종류: 행렬 (Matrix)

행렬(Matrix) 생성: 문자열 행렬 데이터 생성

- 문자 타입 자료(Character)는 " "를 꼭 표시한다.

```
> chars <- c("a", "b", "c", "d", "e", "f", "g", "h", "i", "j") # 문자열 벡터
> mat5 <- matrix(chars) # 문자열 벡터를 '열의 개수가 1개인 행렬'로 만들
> mat5
      [,1]
[1,] "a"
[2,] "b"
[3,] "c"
[4,] "d"
[5,] "e"
[6,] "f"
[7,] "g"
[8,] "h"
[9,] "i"
[10,] "j"
> mat6 <- matrix(chars, nrow = 5) # 문자열 벡터를 '행의 개수가 5개인 5*2 행렬'로 만들
> mat6
      [,1] [,2]
[1,] "a"   "f"
[2,] "b"   "g"
[3,] "c"   "h"
[4,] "d"   "i"
[5,] "e"   "j"
```

데이터 종류: 행렬 (Matrix)

행렬(Matrix) 연산

```
> x1 <- matrix(1:8, nrow=2)
> x1
      [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> x1*3 # 행렬 x1의 각 원소에 3을 곱한다
      [,1] [,2] [,3] [,4]
[1,]    3    9   15   21
[2,]    6   12   18   24
> x1*c(10,20) # 행렬 x1의 첫 번째 행에는 10, 두 번째 행에는 20을 곱한다
      [,1] [,2] [,3] [,4]
[1,]   10   30   50   70
[2,]   40   80  120  160
```

데이터 종류: 행렬 (Matrix)

행렬(Matrix) 연산

- Apply: 행 또는 열에 적용할 수 있는 함수 (Max, Min, Mean, Sum 등)

```
> mat1
      [,1] [,2] [,3]
vec1    1    2    3
vec2    4    5    6
vec3    7    8    9
> apply(mat1, 1, max) # 행렬 mat1의 각 행에서 가장 큰 원소를 출력 (1: 행 방향)
vec1 vec2 vec3
    3    6    9
> apply(mat1, 2, mean) # 행렬 mat1의 각 열의 원소들의 평균
[1] 4 5 6
```

데이터 종류: 행렬 (Matrix)

행렬(Matrix) 연산

```
> x1 <- matrix(1:8, nrow=2); x1
      [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> x2 <- matrix(9:16, nrow=2); x2
      [,1] [,2] [,3] [,4]
[1,]    9   11   13   15
[2,]   10   12   14   16
> x3 <- matrix(9:16, nrow=4); x3
      [,1] [,2]
[1,]    9   13
[2,]   10   14
[3,]   11   15
[4,]   12   16

> x1+x2 # 행렬의 덧셈
      [,1] [,2] [,3] [,4]
[1,]   10   14   18   22
[2,]   12   16   20   24
> x1-x2 # 행렬의 뺄셈
      [,1] [,2] [,3] [,4]
[1,]   -8   -8   -8   -8
[2,]   -8   -8   -8   -8
> x1/x2 # 행렬 각 성분의 나눗셈
      [,1] [,2] [,3] [,4]
[1,] 0.1111111 0.2727273 0.3846154 0.4666667
[2,] 0.2000000 0.3333333 0.4285714 0.5000000
> x1*x2 # 행렬 각 성분의 곱셈
      [,1] [,2] [,3] [,4]
[1,]    9   33   65  105
[2,]   20   48   84  128
> x1 %*% x3 # 행렬의 곱셈
      [,1] [,2]
[1,]   178   242
[2,]   220   300
```

데이터 종류: 배열 (Array)

- 배열(Array) : 행렬을 3차원 이상으로 다차원으로 확장한 것
 - `dim=c(row, column, floor)` parameter를 이용하여 차원 설정 가능

```
> x <- array(1:24, dim=c(2,4,3)) # 3차원 배열 생성 (size: 2*4*3)
> x
, , 1
      [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

, , 2
      [,1] [,2] [,3] [,4]
[1,]    9   11   13   15
[2,]   10   12   14   16

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

| | 1 | 2 | 3 | 4 | |
|---|----|----|----|----|---|
| 1 | 1 | 3 | 5 | 7 | 1 |
| 2 | 2 | 4 | 6 | 8 | |
| 1 | 9 | 11 | 13 | 15 | 2 |
| 2 | 10 | 12 | 14 | 16 | |
| 1 | 17 | 19 | 21 | 23 | 3 |
| 2 | 18 | 20 | 22 | 24 | |

데이터 종류: 배열 (Array)

배열(Array)의 원소 표시

```
> x[1,,] # 1번째 row만을 모음
```

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

```
[1,]    1     9    17
```

```
[2,]    3    11    19
```

```
[3,]    5    13    21
```

```
[4,]    7    15    23
```

```
> mean(x[1,,]) # 그 원소들의 평균
```

```
[1] 12
```

```
> x[1,3,] # 1번째 row & 3번째 column만을 모음
```

```
[1] 5 13 21
```

```
> mean(x[1,3,]) # 그 원소들의 평균
```

```
[1] 13
```

| | 1 | 2 | 3 |
|---|---|----|----|
| 1 | 1 | 9 | 17 |
| 2 | 3 | 11 | 19 |
| 3 | 5 | 13 | 21 |
| 4 | 7 | 15 | 23 |

1번째 row에 속한
12개 원소들
Ex) 1 : (1,1,1) → (1,1)
Ex) 13 : (1,3,2) → (3,2)

데이터 종류: 배열 (Array)

[실습3]

- 다음 코드를 실행한 결과가 무엇일지 생각해봅시다.

```
#Exercise 3
```

```
x <- array(1:24, dim=c(2,4,3)) # 3차원 배열 생성 (size: 2*4*3)
```

```
x[,2,]
```

```
mean(x[,2,])
```

```
x[,2,3]
```

```
x[2,4,3]
```

| | 1 | 2 | 3 | 4 | |
|---|----|----|----|----|---|
| 1 | 1 | 3 | 5 | 7 | 1 |
| 2 | 2 | 4 | 6 | 8 | 1 |
| 1 | 9 | 11 | 13 | 15 | 2 |
| 2 | 10 | 12 | 14 | 16 | 2 |
| 1 | 17 | 19 | 21 | 23 | 3 |
| 2 | 18 | 20 | 22 | 24 | 3 |

데이터 종류: 배열 (Array)

[실습3]

- 다음 코드를 실행한 결과가 무엇일지 생각해봅시다.

```
> x <- array(1:24, dim=c(2,4,3)) # 3차원 배열 생성 (size: 2*4*3)
> x[,2,]
      [,1] [,2] [,3]
[1,]    3   11   19
[2,]    4   12   20
> mean(x[,2,])
[1] 11.5
> x[,2,3]
[1] 19 20
> x[2,4,3]
[1] 24
```

| | 1 | 2 | 3 |
|---|---|----|----|
| 1 | 3 | 11 | 19 |
| 2 | 4 | 12 | 20 |

2번째 column에 속한
6개 원소들
Ex) 4 : (2,2,1) → (2,1)
Ex) 19 : (1,2,3) → (1,3)

1~24 중 제일 마지막 원소 24는
가장 마지막 index (2,4,3)에 배치된다.

데이터 종류: 데이터프레임 (DataFrame)

- 데이터프레임(DataFrame) : 동일하거나 다른 형태의 자료를 2차원 이상의
형태로 여러 개 모아서 취급하는 데이터 형태
 - 보통 N 개의 행과 M 개의 열로 구성되어 구성될 수 있음.
- 데이터프레임 생성 방법
 - 배열 데이터를 (벡터 형태로) 선언한다.
 - 선언된 배열 데이터를 모아서 구성한다.

데이터 종류: 데이터프레임 (DataFrame)

데이터프레임 (DataFrame) 생성

```
> no <- c(1,2,3,4)
> name <- c("Apple", "Banana", "Peach", "Berry")
> prices <- c(500,200,200,50)
> quantity <- c(5,2,7,9)
>
> fruit <- data.frame(No = no, Name = name, PRICE = prices, QTY = quantity)
> #Datframe 생성
> fruit
  No  Name PRICE QTY
1  1 Apple   500   5
2  2 Banana  200   2
3  3 Peach   200   7
4  4 Berry    50   9
```

데이터 종류: 데이터프레임 (DataFrame)

데이터프레임 (DataFrame) 특정 행과 열 출력

```
> fruit[1,] # 1번째 행만 출력
  No  Name PRICE QTY
1  1 Apple   500   5
> fruit[2:4,] # 2~4번째 행 출력
  No  Name PRICE QTY
2  2 Banana  200   2
3  3 Peach   200   7
4  4 Berry    50   9
> fruit[,1] # 1번째 열만 출력
[1] 1 2 3 4
> fruit[, -3] # 3번째 열만 제외하고 출력
  No  Name QTY
1  1 Apple   5
2  2 Banana  2
3  3 Peach   7
4  4 Berry   9

> fruit$No
[1] 1 2 3 4
> fruit$PRICE
[1] 500 200 200 50
> fruit$PRICE * fruit$QTY
[1] 2500 400 1400 450
```

데이터 종류: 리스트 (List)

- 리스트(List) : 키(Key), 값(Value)의 형태로 구성된 데이터
 - 객체의 특정 부분만 호출 및 사용하고자 할 때 편리.
 - 외부에서 Dataset을 Import하면 List형태로 저장됨.

```
> member <- list(name="David",address="Seoul",tel="3370", room=2501) # List 생성
> member
$name
[1] "David"

$address
[1] "Seoul"

$tel
[1] "3370"

$room
[1] 2501

>
> member$name # 'name' Key에 대한 value만 확인
[1] "David"
> member$address # 'address' Key에 대한 value만 확인
[1] "Seoul"
> member$tel # 'tel' Key에 대한 value만 확인
[1] "3370"
> member$room # 'room' Key에 대한 value만 확인
[1] 2501
```

키(Key) : name, address, tel, room
값(Value) : "David", "Seoul", "3370", 2501

List

| key | value |
|-----|-------|
| key | value |
| key | value |
| key | value |

데이터 종류: 리스트 (List)

- 새로운 데이터 (Key & Value) 추가, 변경, 제거

```
> member$birth <- "1999-07-13" # birth Key 추가
> member
$name
[1] "David"

$address
[1] "Seoul"

$tel
[1] "3370"

$room
[1] 2501

$birth
[1] "1999-07-13"
```

```
> member$room <- 3501 # room Key의 value 변경
> member
$name
[1] "David"

$address
[1] "Seoul"

$tel
[1] "3370"

$room
[1] 3501

$birth
[1] "1999-07-13"
```

```
> member$address <- NULL # address Key와 value 제거
> member
$name
[1] "David"

$tel
[1] "3370"

$room
[1] 3501

$birth
[1] "1999-07-13"
```

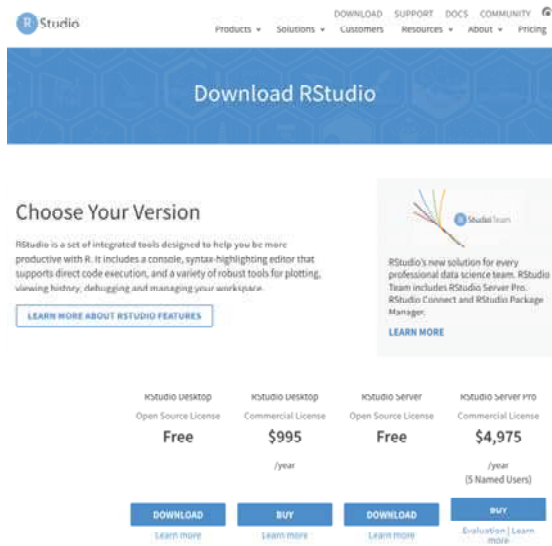

R studio 설치

R studio 설치

- R : 상호 대화식 인터페이스 (Python Jupyter Notebook과 비슷)
 - `_` 명령문에 대한 결과를 곧장 확인 가능하여 편리함.
 - 간단한 코드를 입력하고 이에 대한 결과를 빠르게 확인할 때 유용.
 - 복잡한 프로그래밍을 하기에는 가독성 등의 문제 발생.
- Rstudio : 코드를 먼저 다 구현하고, 결과를 차례대로 확인.
 - 코드 입력창, 결과창(Console), 환경 창, Plot 창 등이 분리되어 있음.
(Python Spyder와 비슷)
 - 복잡한 프로그래밍을 구현하기에 용이함.
 - R을 먼저 설치한 후, Rstudio 설치 가능

R studio 설치

- <https://rstudio.com/products/rstudio/download/>



[Rstudio Desktop] - [DOWNLOAD]

R studio 설치

- 운영체제 선택하여 설치

RStudio Desktop 1.3.1056 - Release Notes

1. Install R: RStudio requires R 3.0.3+.
2. Download RStudio Desktop: Recommended for your system.

DOWNLOAD RSTUDIO FOR MAC
1.3.1056 | 64-bit | 148.84 MB

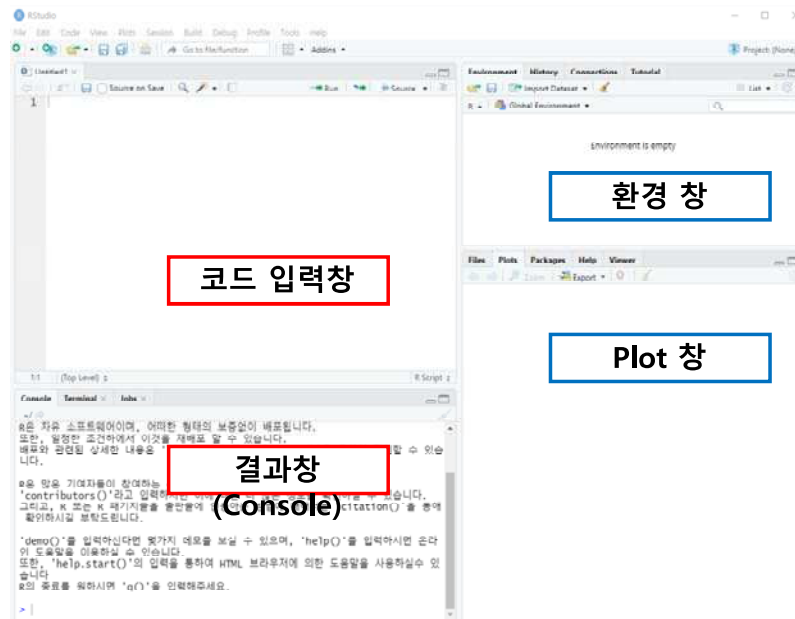
All Installers

Linux users may need to import RStudio's public code signing key prior to installation, depending on the operating system's security policy.
RStudio requires a 64-bit operating system. If you are on a 32-bit system, you can use an older version of RStudio.

| OS | Download | Size | SHA-256 |
|---------------------|---|-----------|-----------|
| Windows 10/8/7 | RStudio-1.3.1056.exe | 171.62 MB | a8f113ae5 |
| macOS 10.13+ | RStudio-1.3.1056.dmg | 148.84 MB | 7383c77d |
| Ubuntu 16 | rstudio-1.3.1056-amd64.deb | 124.56 MB | 00d5e5e5 |
| Ubuntu 14/Debian 10 | rstudio-1.3.1056-amd64.deb | 126.90 MB | 00d5e5e5 |
| Fedora 19/Red Hat 7 | rstudio-1.3.1056-x86_64.rpm | 148.88 MB | 00d5e5e5 |
| Fedora 20/Red Hat 8 | rstudio-1.3.1056-x86_64.rpm | 150.95 MB | 00d5e5e5 |
| Debian 9 | rstudio-1.3.1056-amd64.deb | 126.65 MB | 00d5e5e5 |
| SLES/OpenSUSE 12 | rstudio-1.3.1056-x86_64.rpm | 119.17 MB | 00d5e5e5 |
| OpenSUSE 15 | rstudio-1.3.1056-x86_64.rpm | 128.14 MB | 00d5e5e5 |

R studio 설치

- R studio 실행화면



데이터 다루기

패키지와 함수

- 패키지(Package)란?

- 특정한 목적의 로직들과 코드들의 집합
- 특정 주제에 대하여 완성도가 높고 설계가 잘된 코드들을 제 3자가 이용하기 쉽도록 패키지 형태로 배포
- 함수(Function), built-in 예제 데이터셋, 패키지 사용 방법에 대한 개요 및 설명서, 함수 도움말파일 등으로 구성

- 함수(function)란?

- 특정한 작업을 수행하기 위해 일련의 구문들을 체계적으로 묶은 것
- 간단히 함수를 실행함으로써 특정 작업을 수행할 수 있음

R studio 시작하기: package install

- R을 이용하여 데이터를 읽고, 처리하고 분석하기 위해서는 각 단계를 지원하는 Package를 설치해야 합니다.
- 사용하고자 하는 함수를 Google에 검색하면 쉽게 해당 함수를 제공하는 package가 나오기 때문에 외우실 필요가 없습니다.
- 작업 환경이 바뀔 때마다 처음 한 번만 설치하면 되고, 주로 사용하는 data.table, plyr, dplyr, reshape의 네 가지 package를 기본으로 설치한 뒤 필요한 package를 추가하여 설치하면 됩니다.

```
1 ##Required packages
2 #데이터 읽어오기
3 install.packages('data.table')
4 install.packages('readxl')
5 #데이터 전처리
6 install.packages('plyr')
7 install.packages('dplyr')
8 #데이터 합치기
9 install.packages('bindrcpp')
10 #데이터 형태 바꾸기
11 install.packages('reshape')
12 #데이터 그래프 그리기
13 install.packages('ggplot2')
```

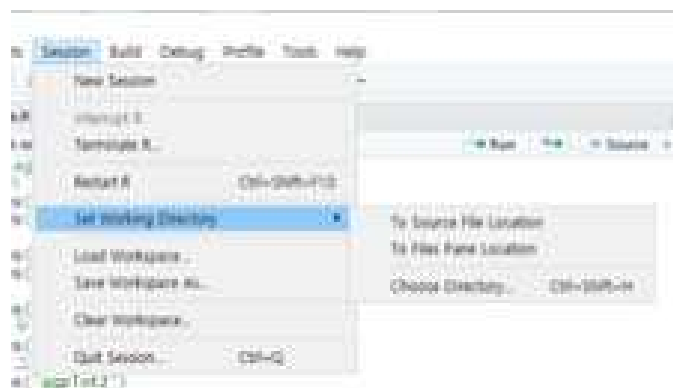
R studio 시작하기: load library

- R package는 한번 설치하면 해당 PC에 저장되어 있습니다.
- 반면 package내의 library를 사용하려면 R studio를 시작할 때 마다 load해주어야 합니다.
- 한번에 library를 모두 load할 필요는 없으며, 필요한 library를 load해가며 분석할 수 있습니다.

```
15 #라이브러리 로드
16 library('data.table')
17 library('readxl')
18 library('plyr')
19 library('dplyr')
20 library('bindrcpp')
21 library('dplyr')
22 library('plyr')
```

R studio 시작하기: 작업 공간 변경

- R에서 기본적으로 데이터를 읽고 쓸 때, 사용할 폴더를 지정해주어야 합니다.
- 서로 다른 폴더에 저장된 데이터도 working directory를 바꾸어가며 작업할 수 있지만, 같은 프로젝트에 대해서는 데이터를 한 폴더에 두는 것이 편리합니다.
- 작업표시줄을 이용하여 working directory를 변경할 수도 있고, 코드를 통해 지정해줄 수도 있습니다.

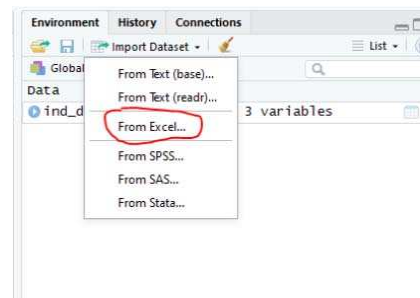


```
24 getwd()
25 setwd('C:/Users/DoHyun-PC/Desktop/r data')
```

R studio 시작하기: 데이터 로드

- 분석할 데이터를 앞서 설정한 working directory 안에 저장하는 것이 편리합니다.
- .xlsx, .csv, .txt, .dat 등 다양한 확장자의 데이터를 사용할 수 있습니다.

```
29 #read_excel 라이브러리를 통한 .xlsx 파일 로드
30
31
32 ind_data <- read_excel('kospi dataset.xlsx',
33                        sheet = 'industry code')
34 prc_data <- read_excel('kospi dataset.xlsx',
35                        sheet = 'stock price')
36
42 #read.csv 라이브러리를 통한 .csv 파일 로드
43
44 ind_data <- read.csv('kospi industry code.csv',
45                    head = TRUE)
46 prc_data <- read.csv('kospi stock price.csv',
47                    head = TRUE)
48
```



데이터 분석하기

코스피 상장 종목 데이터

| | symbol | name | mktcap | industry |
|----|---------|----------|-----------|----------|
| 1 | A005930 | 삼성전자 | 533698560 | 제조업 |
| 2 | A000660 | SK하이닉스 | 98280319 | 제조업 |
| 3 | A051910 | LG화학 | 69886420 | 제조업 |
| 4 | A035420 | NAVER | 57327925 | 제조업 |
| 5 | A005380 | 현대차 | 55553729 | 제조업 |
| 6 | A006400 | 삼성SDI | 54323979 | 제조업 |
| 7 | A207940 | 삼성바이오로직스 | 52270350 | 제조업 |
| 8 | A068270 | 셀트리온 | 43334295 | 제조업 |
| 9 | A035720 | 카카오 | 41905696 | 제조업 |
| 10 | A000270 | 기아차 | 37820400 | 제조업 |
| 11 | A012330 | 현대모비스 | 33031506 | 제조업 |
| 12 | A066570 | LG전자 | 29129311 | 제조업 |

데이터 분석하기

- 데이터 구조 파악

```
> head(ind_data) # 상위 5개 자료 요약
# A tibble: 6 x 4
  symbol name      mktcap industry
  <chr>  <chr>      <dbl>  <chr>
1 A005930 삼성전자  533698560 제조업
2 A000660 SK하이닉스  98280319  제조업
3 A051910 LG화학    69886420  제조업
4 A035420 NAVER      57327925  제조업
5 A005380 현대차     55553729  제조업
6 A006400 삼성SDI     54323979  제조업

> summary(ind_data) # 데이터 요약
      symbol      name      mktcap      industry
Length:779   Length:779   Min.    : 19226   Length:779
Class :character Class :character  1st Qu.: 126498 Class :character
Mode  :character Mode  :character  Median : 289776 Mode  :character
                Mean  : 2708900
                3rd Qu.: 929698
                Max.   :533698560

> ls(ind_data) # 변수명을 abc 순서로 보여줌
[1] "industry" "mktcap"  "name"    "symbol"

> str(ind_data) # 변수의 타입
tibble [779 x 4] (S3: tbl_df/tbl/data.frame)
 $ symbol : chr [1:779] "A005930" "A000660" "A051910" "A035420" ...
 $ name   : chr [1:779] "삼성전자" "SK하이닉스" "LG화학" "NAVER" ...
 $ mktcap : num [1:779] 5.34e+08 9.83e+07 6.99e+07 5.73e+07 5.56e+07 ...
 $ industry: chr [1:779] "제조업" "제조업" "제조업" "제조업" ...

> dim(ind_data) # 변수의 행과 열 개수
[1] 779 4
```

데이터 분석하기

- Dataset 일부를 추출하여 Sub-dataset 만들기 + rbind

```
> data1 <- ind_data[1:2,]; data1 #ind data의 1, 2행을 분리하여 data1로 저장
  symbol      name      mktcap industry
1 A005930 삼성전자  533698560  제조업
2 A000660 SK하이닉스  98280319   제조업

> data2 <- ind_data[3:6,]; data2 #ind data의 3, 4, 5, 6행을 분리하여 data2로 저장
  symbol      name      mktcap industry
1 A005930 삼성전자  533698560  제조업
2 A000660 SK하이닉스  98280319   제조업
3 A051910 LG화학    69886420   제조업
4 A035420 NAVER      57327925   제조업
5 A005380 현대차     55553729   제조업
6 A006400 삼성SDI     54323979   제조업

> newdata1 <- rbind(data1, data2); newdata1 # data1, data2를 행 (row) 방향으로 결합
  symbol      name      mktcap industry
1 A005930 삼성전자  533698560  제조업
2 A000660 SK하이닉스  98280319   제조업
3 A051910 LG화학    69886420   제조업
4 A035420 NAVER      57327925   제조업
5 A005380 현대차     55553729   제조업
6 A006400 삼성SDI     54323979   제조업
```

데이터 분석하기

- Dataset 일부를 추출하여 Sub-dataset 만들기 + cbind

```
> newdata2 <- cbind(data3, data4); newdata2 # data3, data4를 열 (column)
방향으로 결합
```

| | data3 | name | industry |
|----|---------|----------|----------|
| 1 | A005930 | 삼성전자 | 제조업 |
| 2 | A000660 | SK하이닉스 | 제조업 |
| 3 | A051910 | LG화학 | 제조업 |
| 4 | A035420 | NAVER | 제조업 |
| 5 | A005380 | 현대차 | 제조업 |
| 6 | A006400 | 삼성SDI | 제조업 |
| 7 | A207940 | 삼성바이오로직스 | 제조업 |
| 8 | A068270 | 셀트리온 | 제조업 |
| 9 | A035720 | 카카오 | 제조업 |
| 10 | A000270 | 기아차 | 제조업 |

데이터 분석하기

- Apply 연습

| | date | samsung | sk | lg | hyundai | KOSPI |
|----|------------|---------|-------|--------|---------|---------|
| 1 | 2010-01-29 | 15680 | 22750 | 200000 | 113000 | 1602.43 |
| 2 | 2010-02-26 | 14880 | 21000 | 215000 | 115000 | 1594.58 |
| 3 | 2010-03-31 | 16360 | 26700 | 240500 | 115500 | 1692.85 |
| 4 | 2010-04-30 | 16980 | 28400 | 283000 | 137000 | 1741.56 |
| 5 | 2010-05-31 | 15520 | 25150 | 273000 | 140000 | 1641.25 |
| 6 | 2010-06-30 | 15480 | 25050 | 309500 | 144500 | 1698.29 |
| 7 | 2010-07-30 | 16200 | 22500 | 329000 | 149000 | 1759.33 |
| 8 | 2010-08-31 | 15120 | 21100 | 345000 | 141500 | 1742.75 |
| 9 | 2010-09-30 | 15540 | 22150 | 333500 | 153000 | 1872.81 |
| 10 | 2010-10-29 | 14900 | 23150 | 347000 | 170000 | 1882.95 |

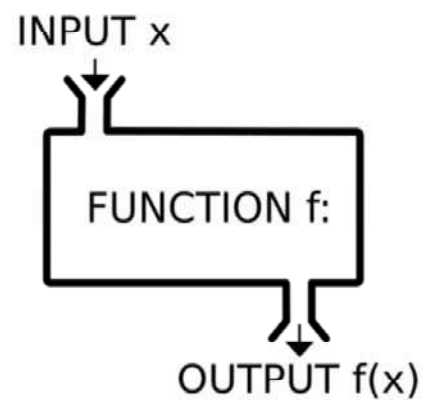
```
> apply(prc_data1, 2, mean) #각 변수(column)별 평균
samsung      sk      lg      hyundai      KOSPI
33782.331 48836.842 339379.699 171812.030 2065.098
> apply(prc_data1, 2, sum) #각 변수(column)별 총합
samsung      sk      lg      hyundai      KOSPI
4493050 6495300 45137500 22851000 274658
> apply(prc_data1, 2, max) #각 변수(column)별 최대값
samsung      sk      lg      hyundai      KOSPI
81000.00 118500.00 824000.00 268500.00 2873.47
> apply(prc_data1, 2, min) #각 변수(column)별 최소값
samsung      sk      lg      hyundai      KOSPI
14880.00 19100.00 181000.00 88700.00 1594.58
```


함수 (Function)

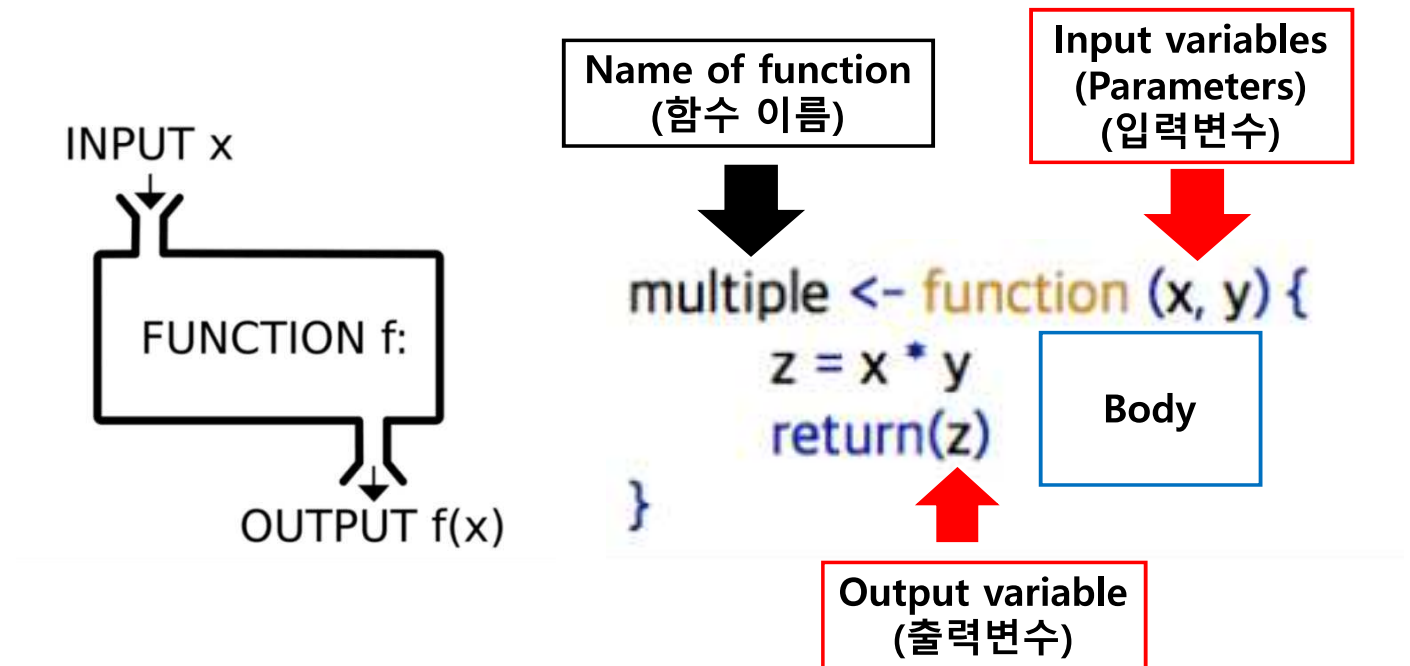
함수 (Function)

- 특정 기능을 반복해서 사용하는 경우, 이 기능을 따로 작성해두고 필요할 때 마다 불러서 사용하면 편리하다.

- 함수(Function) : 특정 기능을 나타내는 코드를 하나의 이름으로 묶은 단위.



함수 (Function)



함수 (Function) 예시

[Example 1] 두 입력변수의 곱을 반환하는 multiple 함수

```
multiple <- function (x, y) {  
  z = x * y  
  return(z)  
}
```

```
> multiple <- function(x,y){  
+   z = x * y  
+   return(z)  
+ }  
> multiple(2,3)  
[1] 6  
> multiple(1.8603,0)  
[1] 0
```

함수 (Function) 예시

[Example 2] 두 수 중 더 큰 수를 출력하는 bigger 함수 (if문 사용)

```
bigger <- function(x, y){  
  if (x>=y) big=x  
  if (x<=y) big=y  
  return(big)  
}
```

```
> bigger <- function(x,y){  
+   if (x>=y) big = x  
+   if (x<=y) big = y  
+   return(big)  
+ }  
> bigger(2,3)  
[1] 3  
> bigger(-0.02, -0.002)  
[1] -0.002
```

함수 (Function) 예시

[연습문제 4] 시간&분&초를 입력하면, 총 몇 초인지를 출력하는 'second' 함수를 만들어 봅시다.

[연습문제 5]

- T시점에서의 주가가 S 인 A 주식을
Strike price(행사가격) $K = \$100$ 에 살 수 있는 권리를 의미하는
Call option의 Payoff는 $P = \max\{S - K, 0\}$ 이며,
그 가격(Option Price; Premium)을 \$5 라 하면
Call option의 Profit은 $P - 5 = \max\{S - K, 0\} - 5$ 입니다.
- T시점에서의 주가가 S 일 때, 이 Call option의 Profit
 $P - 5 = \max\{S - K, 0\} - 5$ 을 계산하는 "callopt" 함수를 만들어봅시다.

함수 (Function) 예시

[연습문제 4] 시간&분&초를 입력하면, 총 몇 초인지를 출력하는 'second' 함수를 만들어봅시다.

```
second <- function(hour, min, sec){  
  second = 3600*hour + 60*min + sec  
  return(second)  
}
```

```
> second <- function(hour, min, sec){  
+   second = 3600 * hour + 60 * min + sec  
+   return(second)  
+ }  
> second(1,20,30)  
[1] 4830  
> x <- second(4,0,1);x  
[1] 14401
```

조건문과 반복문

조건문

- 특정 조건에 따라 수행되는 작업의 내용이 다른 경우.

Ex1) 특정 범위 안에 속하는 수에 대해서는 1, 그 외에는 0을 주는 Indicator function.

Ex2) Call option & Put option의 Payoff

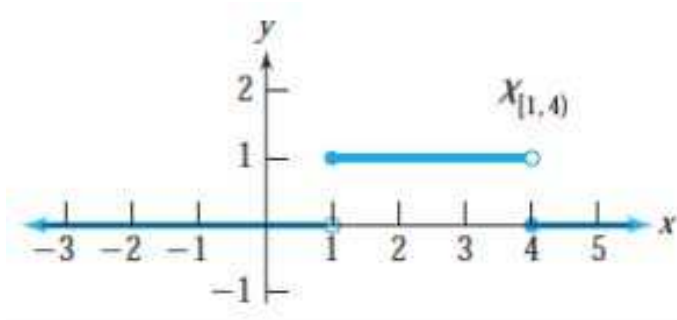
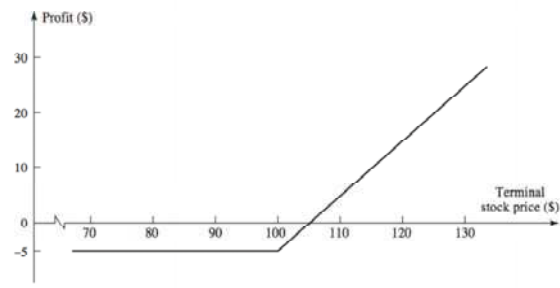


Figure 10.1 Profit from buying a European call option on one share of a stock. Option price = \$5; strike price = \$100.



조건문

- 조건문의 종류

- If (조건) 명령문

- 조건에 맞으면 명령문 수행 (가장 기본적인 형태의 조건문)

- If (조건) 명령문1 else 명령문2

- 조건에 맞으면 명령문1 수행, 맞지 않으면 명령문2 수행

- ifelse (조건, 명령문1, 명령문2)

- 조건에 맞으면 명령문1 수행, 맞지 않으면 명령문2 수행

- switch (기준, 조건1=명령문1, 조건2=명령문2,)

- 기준을 정하고 조건1이 발생하면 명령문1 수행,
조건2가 발생하면 명령문2 수행.

조건문

- If (조건) 명령문

```
> x <- runif(1); x # 0과 1 사이의 난수를 생성해서 x에 저장
[1] 0.8399932
> if (x > 0.5){
+   print(x+2)
+ } #x가 0.5보다 크면 x+2 출력
[1] 2.839993
> x <- runif(1); x # 0과 1 사이의 난수를 생성해서 x에 저장
[1] 0.03954729
> if (x > 0.5){
+   print(x+2)
+ } #x가 0.5보다 크면 x+2 출력
```

조건문

- If (조건) 명령문1 else 명령문2

```
> x <- runif(1); x # 0과 1 사이의 난수를 생성해서 x에 저장
[1] 0.9597219
> if (x > 0.5){
+   print("Large")
+ } else {
+   print("Small")
+ } #x가 0.5보다 크면 Large 출력, 0.5보다 작으면 Small 출력
[1] "Large"
> x <- runif(1); x # 0과 1 사이의 난수를 생성해서 x에 저장
[1] 0.2900413
> if (x > 0.5){
+   print("Large")
+ } else {
+   print("Small")
+ } #x가 0.5보다 크면 Large 출력, 0.5보다 작으면 Small 출력
[1] "Small"
```

조건문

- Ifelse (조건, 명령문1, 명령문2)

```
> x <- runif(1); x # 0과 1 사이의 난수를 생성해서 x에 저장
[1] 0.7403247
> ifelse(x>0.5, "Large", "Small") #x가 0.5보다 크면 Large 출력, 0.5보다 작으면 Small 출력
[1] "Large"
> x <- runif(1); x # 0과 1 사이의 난수를 생성해서 x에 저장
[1] 0.254151
> ifelse(x>0.5, "Large", "Small") #x가 0.5보다 크면 Large 출력, 0.5보다 작으면 Small 출력
[1] "Small"
```

조건문

[연습문제 5]

- T시점에서의 주가가 S 인 A 주식을
Strike price(행사가격) $K = \$100$ 에 살 수 있는 권리를 의미하는
Call option의 Payoff는 $P = \max\{S - K, 0\}$ 이며,
그 가격(Option Price; Premium)을 \$5 라 하면
Call option의 Profit은 $P - 5 = \max\{S - K, 0\} - 5$ 입니다.
- T시점에서의 주가가 S 일 때, 이 Call option의 Profit
 $P - 5 = \max\{S - K, 0\} - 5$ 을 계산하는 "callopt" 함수를 만들어봅시다.

조건문

[연습문제 5]

- T시점에서의 주가가 S 일 때, 이 Call option의 Profit

$P - 5 = \text{Max} \{S - K, 0\} - 5$ 을 계산하는 "callopt" 함수를 만들어봅시다.

```
callopt <- function(S) {  
  K = 100  
  profit <- ifelse(S >= K, S - K - 5, -5)  
  return(profit)  
}
```

```
> callopt(110) # Profit : (110-100)-5 = $5  
[1] 5  
> callopt(105) # Profit : (105-100)-5 = $0  
[1] 0  
> callopt(100) # Profit : (100-100)-5 = $-5  
[1] -5  
> callopt(90) # Profit : -5 = $-5  
[1] -5
```

반복문

- 특정 작업이 반복되는 경우, 그 반복을 실행하는 구문.
- **프로그래밍 기능의 핵심**: 코드를 효율적으로 작성하는 데 꼭 필요.
- 반복문의 종류
 - for (반복) : 특정 (반복) 대상에 대해 Loop를 계속 수행.
 - while (조건) : 특정 (조건)을 만족하면 Loop를 계속 수행.
 - repeat + Break : 일단 실행하고, 특정 조건서 반복문 탈출.
 - C에서의 "do-while"문과 비슷하다.

반복문

- 특정 작업이 반복되는 경우, 그 반복을 실행하는 구문.
- **프로그래밍 기능의 핵심**: 코드를 효율적으로 작성하는 데 꼭 필요.
- 반복문의 종류
 - for (반복) : 특정 (반복) 대상에 대해 Loop를 계속 수행.
 - while (조건) : 특정 (조건)을 만족하면 Loop를 계속 수행.
 - repeat + Break : 일단 실행하고, 특정 조건서 반복문 탈출.
 - C에서의 "do-while"문과 비슷하다.

반복문

- for (**반복**) 반복문

```
> #반복문
> sum1 <- 0
> for (i in 1:10){
+   sum1 <- sum1 + i
+   print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
> sum1
[1] 55

> sum2<-0
> for (i in 1:5){
+   for (j in 1:5){
+     sum2 <- sum2+i*j
+   }
+ }
> sum2
[1] 225
```

반복문

- while (조건) 반복문

```
> sum3 <- 0
> i <- 0
> while (i<=10){ # i가 10 이하면 {}안의 구문 실행.
+   sum3 <- sum3 + i
+   i <- i+1 # i가 1씩 증가하므로 언젠가는 반복문이 끝난다.
+ }
> sum3
[1] 55
```

반복문

[연습문제 6]

- 11에서 200까지의 자연수 중, 소수(Prime number; 1과 자기 자신만을 약수로 가지는 자연수)들을 모은 벡터(Vector) "Prime"을 만들어 봅시다.
- Hint
 - 아무 원소도 없는 벡터는 `c()`로 정의 가능합니다.
 - $\forall i \in \mathbb{N}$ m 이 0과 같은지 아닌지를 나타내는 코드는 `"if (m==0)"`입니다.
 - 벡터 x 에 b 라는 원소를 새로 추가하려면 `"x <- c(x, b)"`를 사용합니다.
 - `i %% j == 0`

[연습문제 6]

- 11에서 200까지의 자연수 중, 소수(Prime number; 1과 자기 자신만을 약수로 가지는 자연수)들을 모은 벡터(Vector) "Prime"을 만들어 봅시다.

```
> prime <- c()
>
> for (i in 11:200){ # 11~200의 자연수에 대한 반복문
+   isprime <- 1 # i가 소수인지 확인하기 위한 더미 변수
+   for (j in 2:(i-1)){ # 2보다 크고 i보다 작은 자연수 j에 대해
+     if(i %% j == 0){ # i를 j로 나눈 나머지가 0이라면
+       isprime <- 0 # isprime을 0으로 바꾸고 (소수가 아님)
+       break # j에 대한 반복문을 탈출한다.
+     }
+   }
+   if (isprime == 1){prime <- c(prime, i)} #isprime이 그대로 1이라면 소수
# 의미하므로 prime vector에 원소로 추가함.
+ } # i에 대한 반복문 끝
>
> prime # 결과 확인
[1] 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71
[17] 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151
[33] 157 163 167 173 179 181 191 193 197 199
> length(prime) # 정답은 42개
[1] 42
```