

R 프로그래밍을 활용한 통계분석

2022 DFMBA 사전교육 – R 프로그래밍

TA 조상흠

1. 통계분석과 R 프로그래밍

통계분석?

- 만약 여러분이 펀드매니저라면?
 - Asset allocation!
 - 연초에 strategic asset allocation
 - 주식, 채권, 크레딧, 통화, 원자재
 - (그리고 가상화폐)에 어떻게 자산을 배분할 것인가!
 - 연중에 tactical asset allocation
 - 거시적 여건과 시장상황에 따라서 포트폴리오를 적극적으로 관리
 - 연말에 성과평가
 - 그렇다면 이런 흐름으로 실제 운용하려면 무엇이 필요하죠?

통계분석?

- 그렇다면 이런 흐름으로 실제 운용하려면 무엇이 필요하죠?

Cliff's Perspectives, Dec 2020

고려 사항	통계학적 해석
Asset class	Data (표본)
기대수익률	평균
리스크 (변동성)	분산
다른 자산과의 동조성	공분산
정책, 이자율, 인플레이션, 어닝쇼크 등에 대한 영향	회귀분석
성과평가	시각화, table 작업을 통한 communication! (제일 중요)

	L/S Mom	L/S Value	L/S Quality	L/S BAB	Portfolio
January – October	2.09	-4.23	1.57	-0.18	-2.21
November – December	-2.62	1.78	-2.95	-1.55	-1.56
Full YTD	1.01	-3.31	0.41	-0.72	-2.62

Source: AQR, Bloomberg. January 1, 2002 – December 15, 2020. Based on daily returns of the Dow Jones indices and weights listed above, portfolio rebalanced daily. Please see index descriptions in the Appendix. Past performance is not a guarantee of future performance. For illustrative purposes only and not representative of a portfolio AQR currently manages. Hypothetical data has inherent limitations, some of which are in the Disclosures.

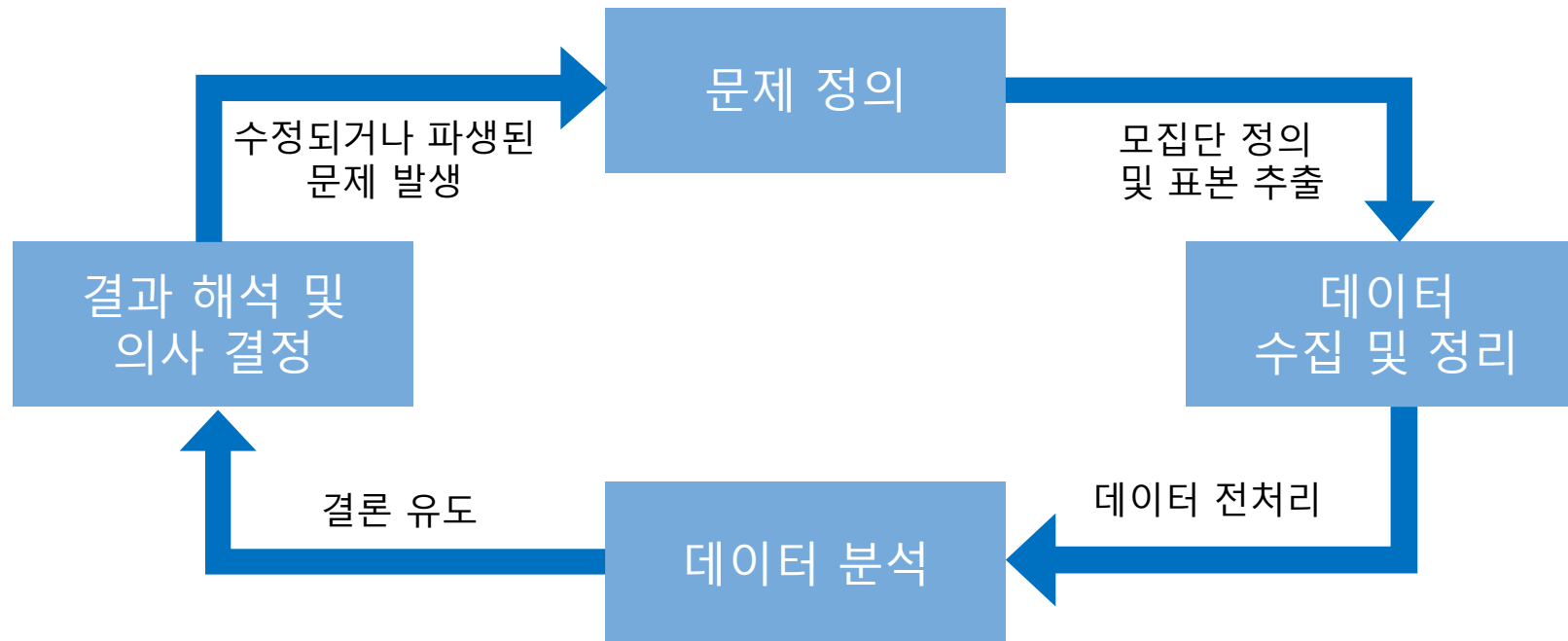
Cumulative Excess Return of the Four Factor Long-Short Portfolio
January 1, 2002 – December 15, 2020



Source: AQR, Bloomberg. January 1, 2002 – December 15, 2020. Based on daily returns of the Dow Jones indices and weights listed above, portfolio rebalanced daily. Please see index descriptions in the Appendix. Past performance is not a guarantee of future performance. For illustrative purposes only and not representative of a portfolio AQR currently manages. Hypothetical data has inherent limitations, some of which are in the Disclosures.

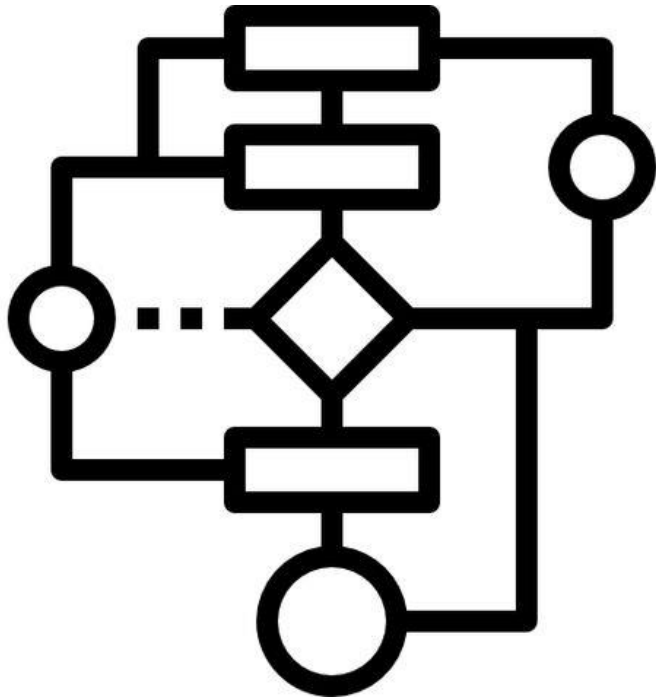
통계분석 도구로서의 R 프로그래밍

- 통계분석?
 - 특정 집단을 대상으로 자료를 수집하여 대상 집단에 대한 정보를 구하고, 적절한 통계분석 방법론을 이용하여 **의사결정 (통계적 추론)**을 하는 과정
- R: 통계분석을 위한 프로그램 언어!



프로그래밍이란?

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



- 예제) Call option 가치 구하기
 - Payoff 계산
 - Input
 - Strike Price (행사가격) : K
 - Future stock price : S_T
 - 연산
 - Calculate $S_T - K$
 - Calculate $\text{Max}(S_T - K, 0)$
 - 사용된 연산
 - $-, \text{Max}()$

프로그래밍이란?

- 프로그래밍 언어
 - 알고리즘을 구현할 수 있는 도구
- 용도
 - 데이터 처리, 데이터 저장 (database), 여러가지 알고리즘 구현, 통계분석 등
- 통계분석으로 사용할 만한 언어들
 - R, python, matlab, **Stata**, SAS
 - (Stata: academic이나 보고서용의 table 만들기의 최강자)
 - (요즘의 panel data analysis는 다 Stata로 넘어오는 추세)

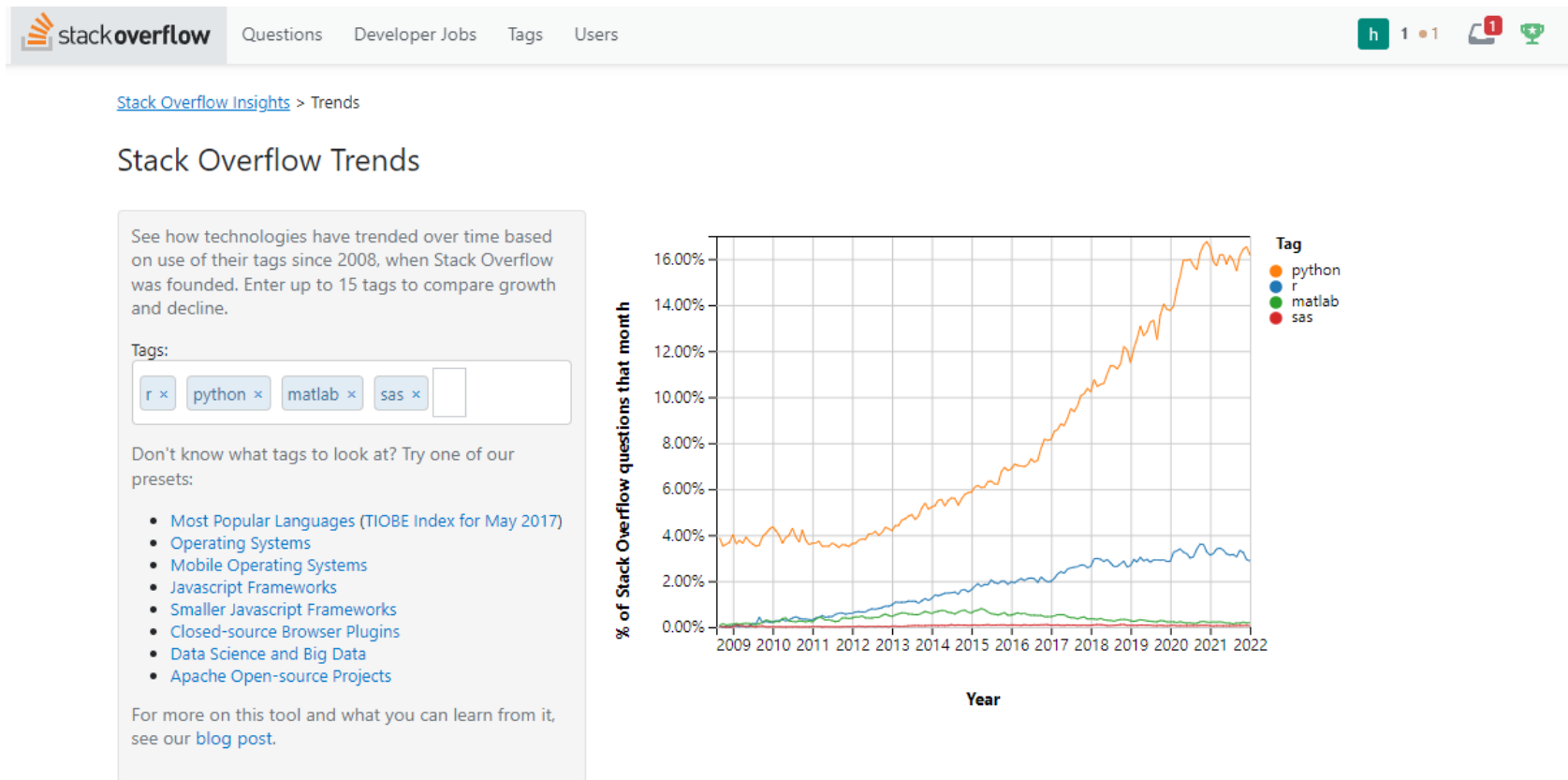


프로그래밍이란?

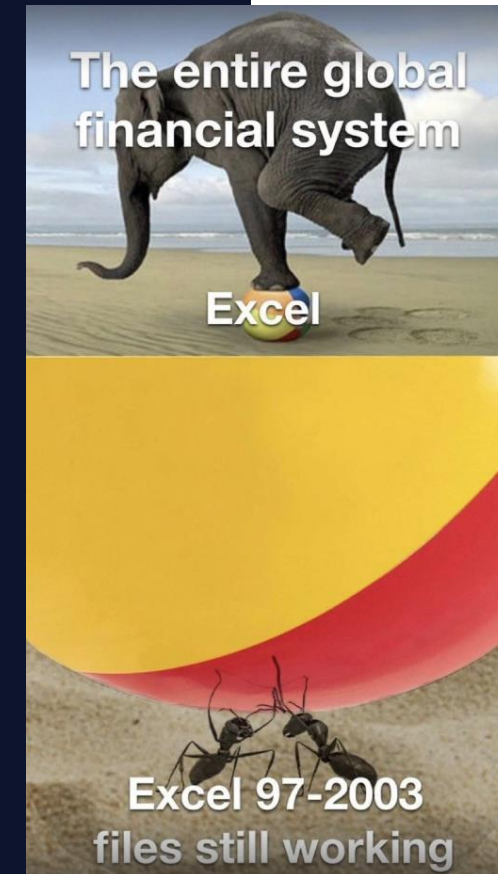
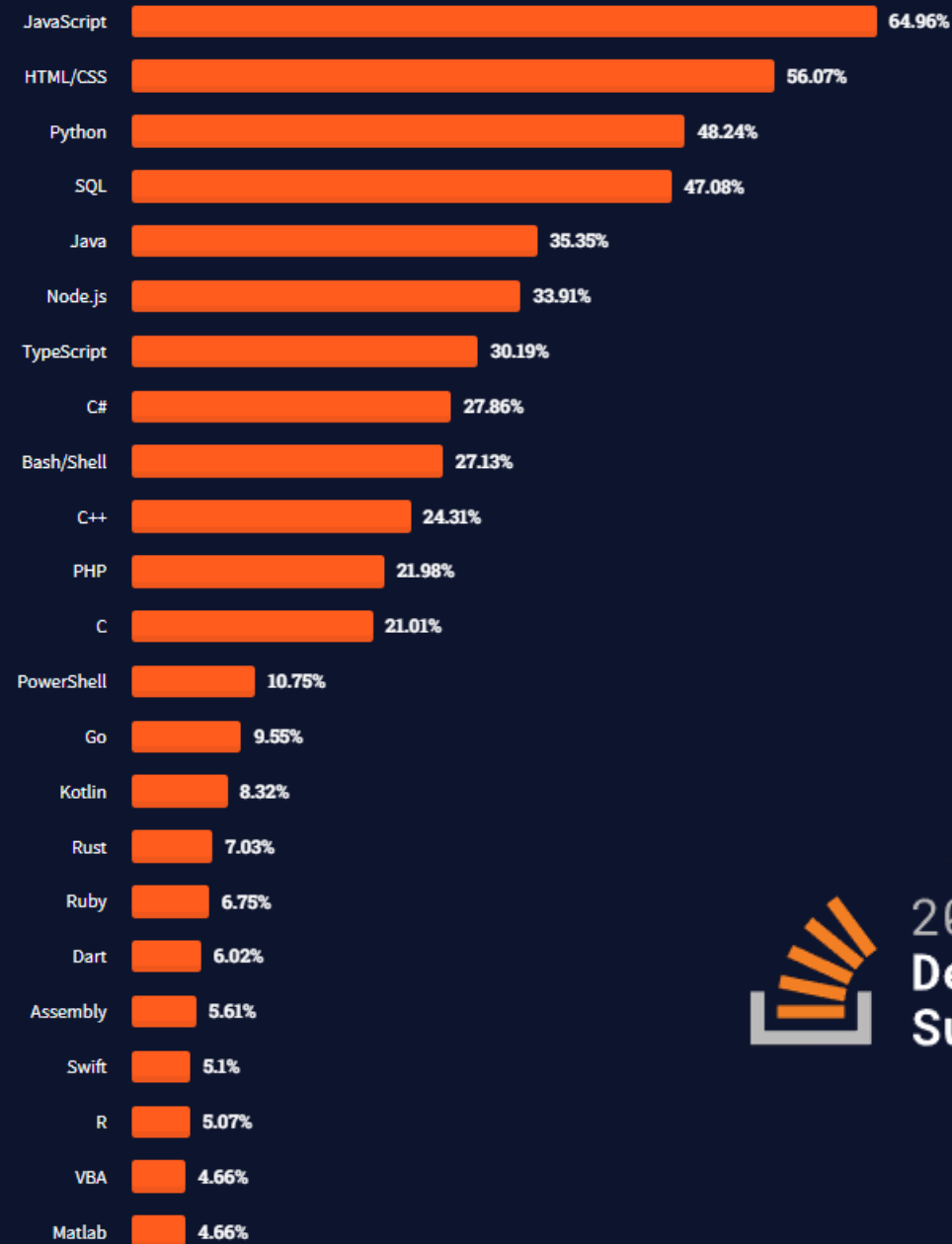
- 프로그래밍 언어 동향
 - 매우 중요!
 - 앞으로 많이 사용할 **언어**를 배우는 것이 매우 중요!
 - Eviews?
 - 과거 시계열 분석의 강자
 - 그런데 지금은?
 - Unit root 검증을 위해 시계열에 복잡한 모델을 적용하는 것 보다 아주 풍부한 panel data를 사용한 **빅데이터 분석**이 중요해짐
 - 즉, 통계모형으로 승부를 보는 시대는 가고 **통계 모형은 간단하지만 다양한 데이터로 승부를 보는 시대**가 도래.
 - 프로그래밍 언어는 트렌드가 매우 중요
 - 매우 빠르고 역동적으로 바뀌니까 시대에 뒤쳐지지 않기 위해서는 잘 tracking해야함.

프로그래밍 트렌드: Stackoverflow Trend and Survey

- <https://insights.stackoverflow.com/trends?tags=r%2Cpython%2Cmatlab%2Csas>
- <https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language>



[Download SVG](#)



프로그래밍 트렌드: Stackoverflow Trend and Survey, including C, C#

[Stack Overflow Insights](#) > Trends

Stack Overflow Trends

See how technologies have trended over time based on use of their tags since 2008, when Stack Overflow was founded. Enter up to 15 tags to compare growth and decline.

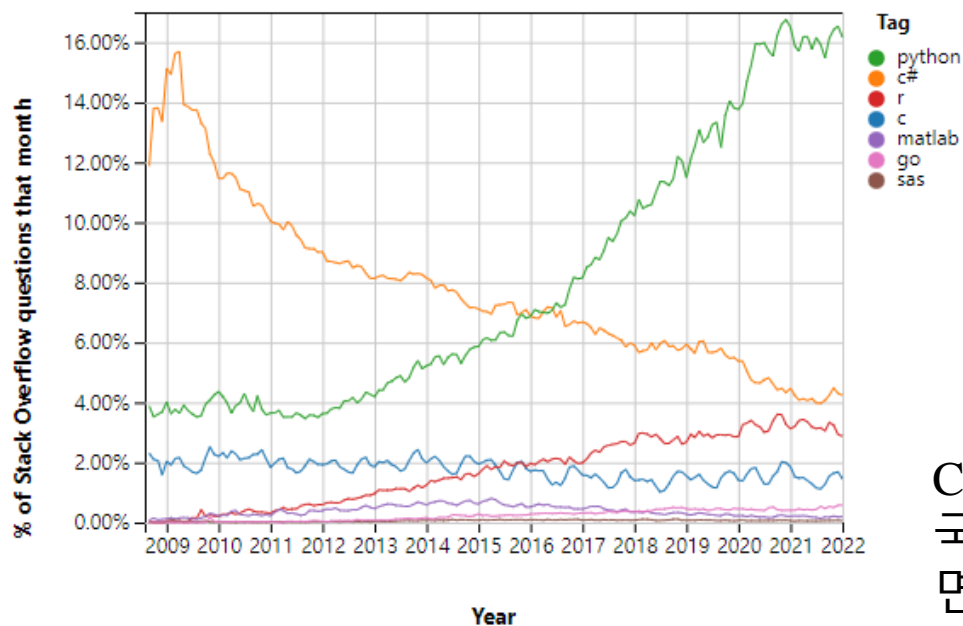
Tags:

c x c# x python x r x matlab x sas x
go x

Don't know what tags to look at? Try one of our presets:

- Most Popular Languages (TIOBE Index for May 2017)
- Operating Systems
- Mobile Operating Systems
- Javascript Frameworks
- Smaller Javascript Frameworks
- Closed-source Browser Plugins
- Data Science and Big Data
- Apache Open-source Projects

For more on this tool and what you can learn from it, see our [blog post](#).



C에 있는 계절성은 미국 가을학기 개강하면서 CS101의 C에 대한 수요?

프로그래밍 트렌드: Stackoverflow Trend and Survey

[Stack Overflow Insights](#) > Trends

Stack Overflow Trends

See how technologies have trended over time based on use of their tags since 2008, when Stack Overflow was founded. Enter up to 15 tags to compare growth and decline.

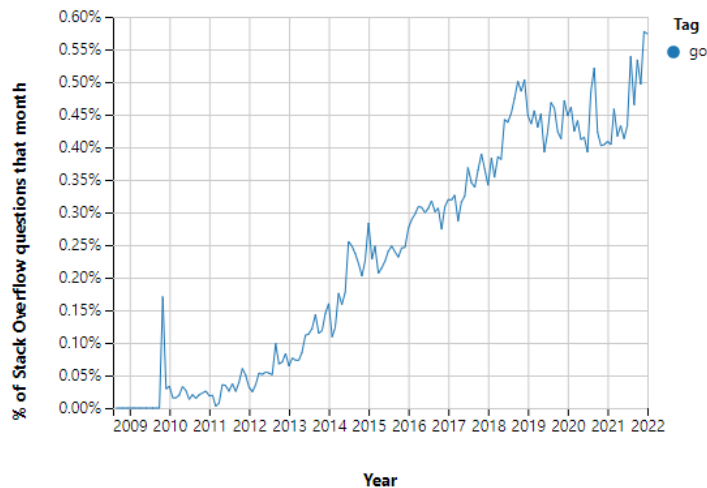
Tags:

go ×

Don't know what tags to look at? Try one of our presets:

- [Most Popular Languages \(TIOBE Index for May 2017\)](#)
- [Operating Systems](#)
- [Mobile Operating Systems](#)
- [Javascript Frameworks](#)
- [Smaller Javascript Frameworks](#)
- [Closed-source Browser Plugins](#)
- [Data Science and Big Data](#)
- [Apache Open-source Projects](#)

For more on this tool and what you can learn from it, see our [blog post](#).



[Stack Overflow Insights](#) > Trends

Stack Overflow Trends

See how technologies have trended over time based on use of their tags since 2008, when Stack Overflow was founded. Enter up to 15 tags to compare growth and decline.

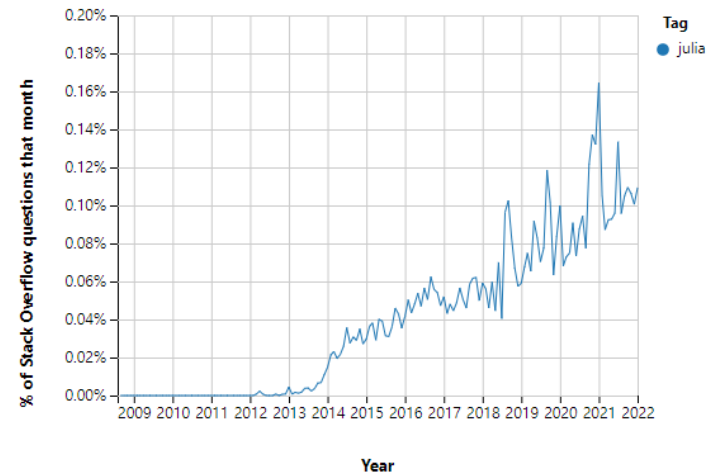
Tags:

julia ×

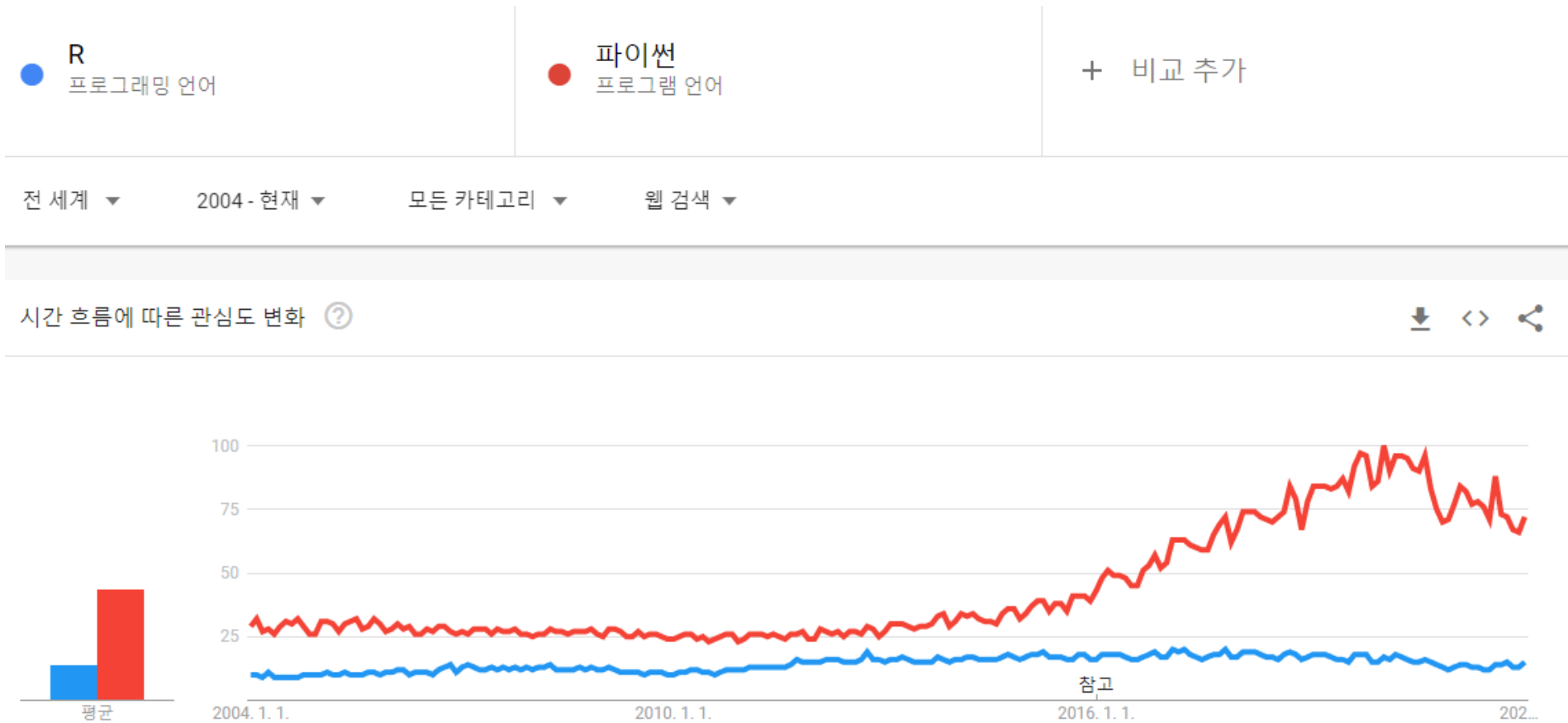
Don't know what tags to look at? Try one of our presets:

- [Most Popular Languages \(TIOBE Index for May 2017\)](#)
- [Operating Systems](#)
- [Mobile Operating Systems](#)
- [Javascript Frameworks](#)
- [Smaller Javascript Frameworks](#)
- [Closed-source Browser Plugins](#)
- [Data Science and Big Data](#)
- [Apache Open-source Projects](#)

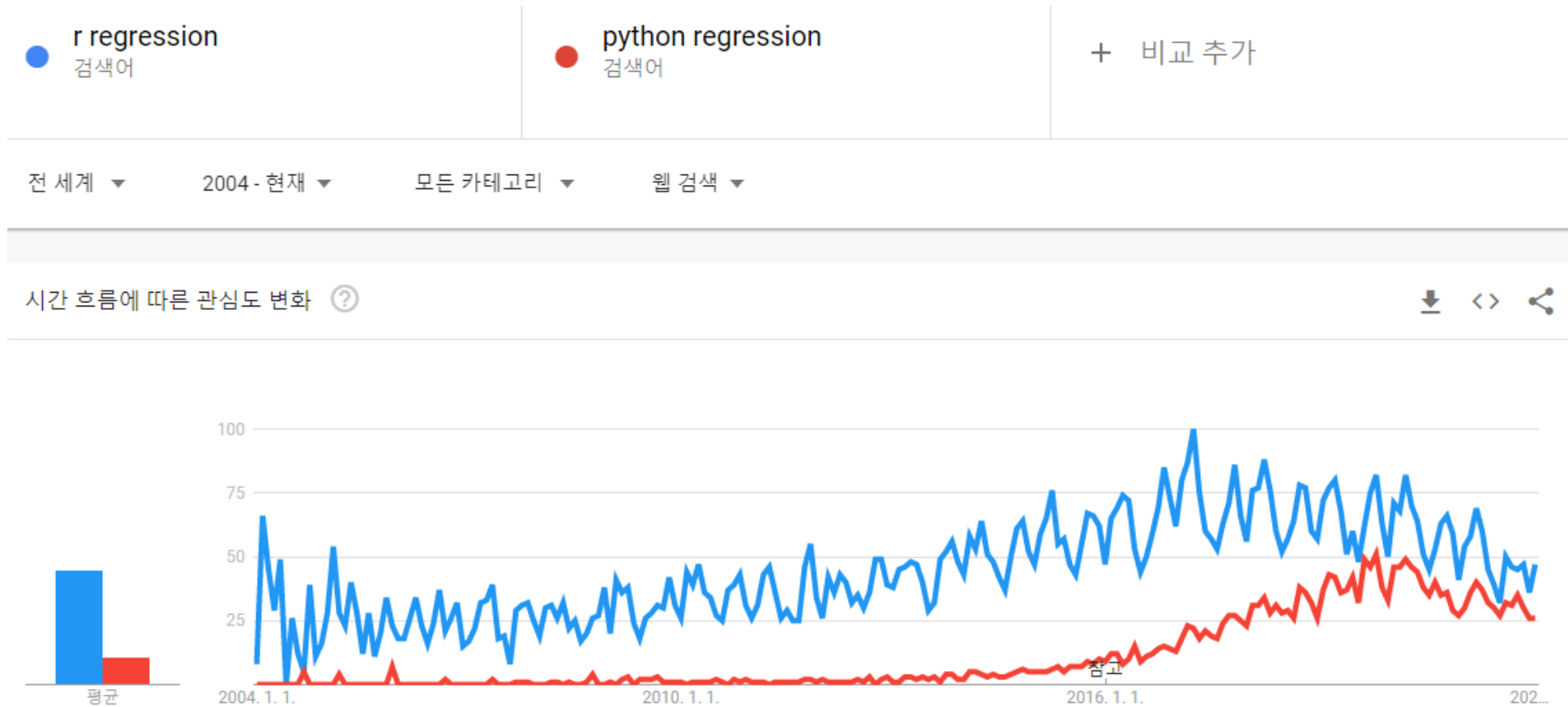
For more on this tool and what you can learn from it, see our [blog post](#).



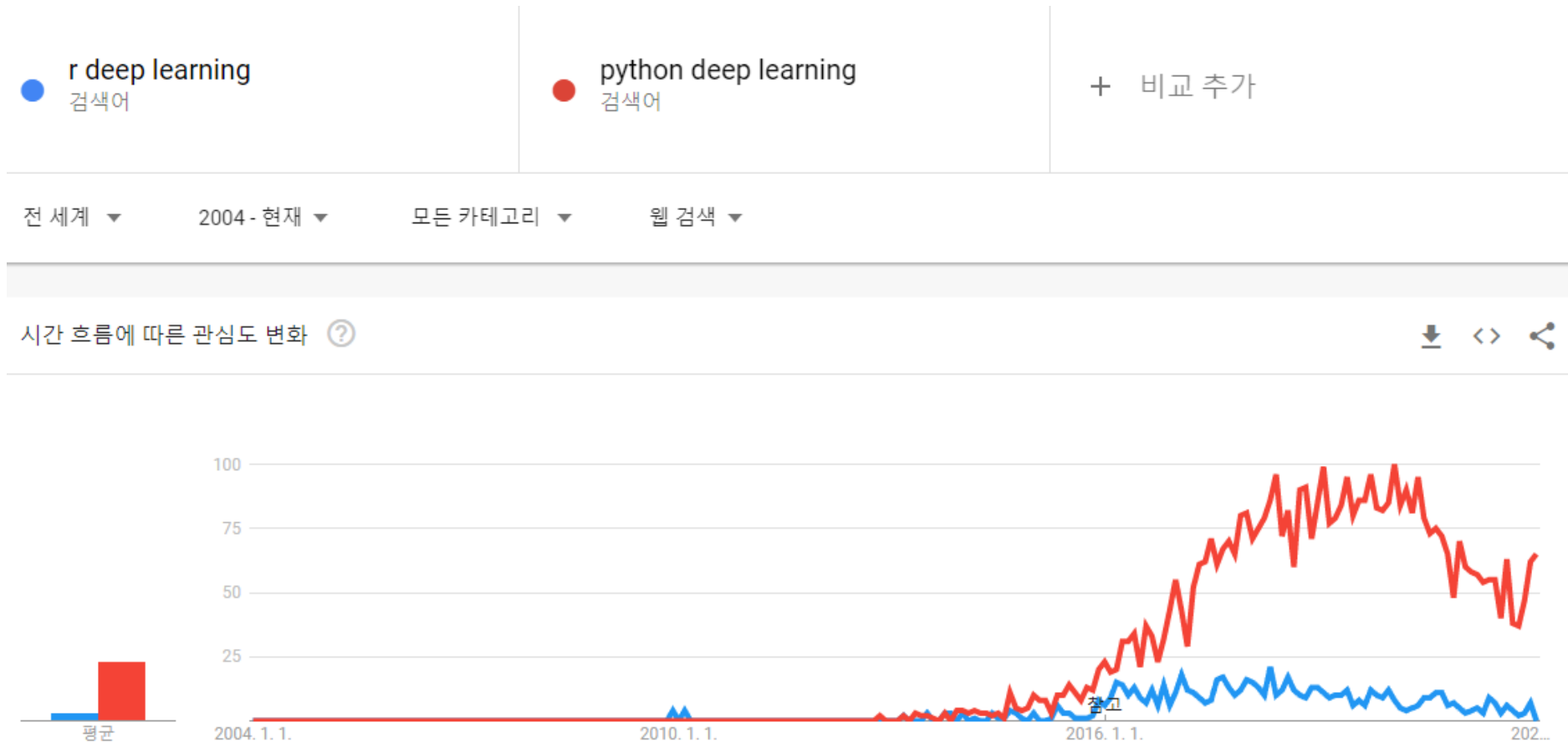
R vs Python: 평균적 트렌드 변화



R vs. Python: 통계분석(regression)과의 연관성 변화



R vs. Python: Deep learning (4차산업혁명..?)과의 연관성 변화



R vs Python

- R

- (Academic) 통계분석

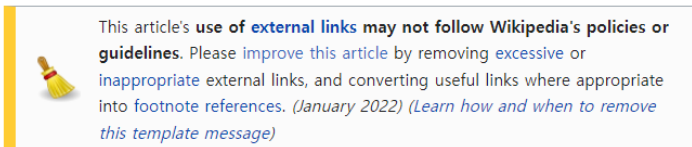
- 핵심 패키지

- tidyverse

- dplyr ggplot ...

Tidyverse

From Wikipedia, the free encyclopedia



The **tidyverse** is a collection of [open source R packages](#) introduced by [Hadley Wickham](#)^[1] and his team that "share an underlying design philosophy, grammar, and data structures" of tidy data.^[2] Characteristic features of tidyverse packages include extensive use of non-standard evaluation and encouraging [piping](#).^{[3][4][5]}

As of November 2018, the tidyverse package and some of its individual packages comprise 5 out of the top 10 most downloaded R packages.^[6] The tidyverse is the subject of multiple books and papers.^{[7][8][9][10]} In 2019, the ecosystem has been published in the *Journal of Open Source Software*.^[11]

Critics of the tidyverse have argued it promotes tools that are harder to teach and learn than their base-R equivalents and are too dis-similar to other programming languages.^{[12][13]}

Contents [hide]

- 1 Packages
- 2 See also
- 3 References
- 4 External links

- Python

- 그 외

- (데이터분석을 위한) 핵심패키지

- pandas

pandas (software)

From Wikipedia, the free encyclopedia

*Not to be confused with **PANDAS**, the Australian archival management system used for the Pandora Archive.*

pandas is a [software library](#) written for the [Python programming language](#) for data manipulation and [analysis](#). In particular, it offers [data structures](#) and operations for manipulating numerical tables and time series. It is [free software](#) released under the [three-clause BSD license](#).^[2] The name is derived from the term "panel data", an [econometrics](#) term for [data sets](#) that include observations over multiple time periods for the same individuals.^[3] Its name is a play on the phrase "Python data analysis" itself.^[4] [Wes McKinney](#) started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.^[5]

Contents [hide]

- 1 Library features
- 2 Dataframes
- 3 History
 - 3.1 Timeline:^[13]
- 4 See also
- 5 References
- 6 Further reading
- 7 External links

pandas



Original author(s)	Wes McKinney
Developer(s)	Community
Initial release	11 January 2008; 14 years ago ^[citation needed]
Stable release	1.4.0 ^[1] / 22 January 2022; 16 days ago
Repository	github.com/pandas-dev/pandas [ⓘ]
Written in	Python, Cython, C
Operating system	Cross-platform
Type	Technical computing
License	New BSD License
Website	pandas.pydata.org [ⓘ]

- benchmarking the R *data.frame* by financial data scientist and very successful! 16

R의 장점

- 통계 분석과 시각화를 제공하는 **무료 프로그램! Free Software!** (향후 발전가능성)
 - **매우 (매우) 중요!**
 - 폭넓은 사용자층: 일반인 to **전문통계학자**: 통계학의 발전을 leading
 - 풍부한 프로그램 소스: **Open source**
- 다양한 운영체제(Windows, MacOS, Linux) 에서 사용 가능 (호환성)
- 다양한 통계 프로젝트를 손쉽게 수행 가능! (범용성)
 - 금융, 기계학습, 데이터마이닝, 경제학, 심리학
- 직관적인 코드



프로그래밍 언어 학습에 대한 노력 배분

- Python
 - 이미 잘 “분산”된 언어
 - 수많은 package들이 스스로 추가수요를 창출하고 그에 맞춰 전문적인 개발인력 (big tech)들이 갈수록 양질의 패키지를 공급
 - Big tech의 2020 R&D 투자: 160조원 (미국 public, private 모두 합치면 약 600조)
 - 요즘, 그리고 앞으로 최고 실력의 CS나 stat 출신이 Google을 갈까 SAS를 갈까?
- R
 - 통계에 집중투자
 - State-of-the-art (academic) 통계기법은 R을 통해서 많이 나옴 (단, AI기술은 모두 python)
 - 기존의 최고 실력 통계학자이나 econometrician들은 주로 R로 코드를 작성하여 소통
- R에 집중투자는 위험. Python 80%, R 20% 정도로 투자하는 것이 바람직!
 - **통계 패키지는 *아직까지는* 의문의 여지없이 R에 더 잘 구현되어 있으니, R을 벤치마크 삼아서 공부하고 실무적으로 python에서 해당 기능을 어떻게 구현할지 고민하면 제일 이상적.**

프로그래밍을 위한 유용한 사이트들

- R Foundation
 - <http://www.r-project.org>
 - R에 관한 여러 정보들
- CRAN (The Comprehensive R Archive Network)
 - <http://cran.r-project.org>
 - 패키지(Packages)들 및 자료 다운로드
- GitHub
 - <https://github.com/>
 - 수많은 오픈소스 패키지들
- Google
 - Stackoverflow or blogs...
 - Google, google, google, (코딩은 구글링의 연속이다!!)



2. R 설치

R 설치

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



[\[Home\]](#)

Download

[CRAN](#)

R Project

[About R](#)

[Logo](#)

[Contributors](#)

[What's New?](#)

[Reporting Bugs](#)

[Conferences](#)

[Search](#)

[Get Involved: Mailing Lists](#)

[Get Involved: Contributing](#)

[Developer Pages](#)

[R Blog](#)

The R Project for Statistical Computing

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 4.1.2 \(Bird Hippie\)](#) has been released on 2021-11-01.
- [R version 4.0.5 \(Shake and Throw\)](#) was released on 2021-03-31.
- Thanks to the organisers of useR! 2020 for a successful online conference. Recorded tutorials and talks from the conference are available on the [R Consortium YouTube channel](#).
- You can support the R Foundation with a renewable subscription as a [supporting member](#)

News via Twitter

R 설치

- CRAN Mirror 고르기

Italy

<https://cran.mirror.garr.it/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://ftp.harukasan.org/CRAN/>

<https://cran.yu.ac.kr/>

<https://cran.seoul.go.kr/>

<https://cran.biodisk.org/>

Information and Database Systems Laboratory, Pukyong National University

Yeungnam University

Bigdata Campus, Seoul Metropolitan Government

The Genome Institute of UNIST (Ulsan National Institute of Science and Technology)

Malaysia

<https://mirrors.upm.edu.my/CRAN/>

Universiti Putra Malaysia

Mexico

<https://cran.itam.mx/>

<https://www.est.colpos.mx/>

Instituto Tecnológico Autónomo de México

Colegio de Postgraduados, Texcoco

Morocco

<https://mirror.marwan.ma/cran/>

MARWAN

Netherlands

<https://mirror.lyrahosting.com/CRAN/>

Lyra Hosting

New Zealand

<https://cran.stat.auckland.ac.nz/>

University of Auckland

Norway

<https://cran.uib.no/>

University of Bergen

R 설치

- 자신에게 맞는 운영체제 선택하여 설치

The Comprehensive R Archive Network

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 (2021-11-01, Bird Hippie) [R-4.1.2.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

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 \geq 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 and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

R 설치: Windows

R-4.1.2 for Windows (32/64 bit)

[Download R 4.1.2 for Windows](#) (86 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 .exe to the [fingerprint](#) on the master server. You will need a version of md5sum for windows: both [graphical](#) 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

- Patches to this release are incorporated in the [r-patched snapshot build](#).
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#).
- [Previous releases](#)

Note to webmasters: A stable link which will redirect to the current Windows binary release is <http://<CRAN MIRROR>/bin/windows/base/release.html>.

Last change: 2021-11-01

R 설치: macOS

R for macOS

This directory contains binaries for a base distribution and packages to run on macOS. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the [old](#) directory.

Note: 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 4.1.2 "Bird Hippie" released on 2021/11/01

Please check the SHA1 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
`openssl sha1 R-4.1.2.pkg`
in the *Terminal* application to print the SHA1 checksum for the R-4.1.2.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
`pkgutil --check-signature R-4.1.2.pkg`

Latest release:

[R-4.1.2.pkg](#) (notarized and signed)
SHA1-hash: 61d3909bc070f7fb86c5a2bd67209fda9408faaa
(ca. 87MB)

R 4.1.2 binary for macOS 10.13 (**High Sierra**) and higher, **Intel 64-bit** build, signed and notarized package.

Contains R 4.1.2 framework, Rapp GUI 1.77 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 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 `tcltk` R package or build package documentation from sources.

Note: the use of X11 (including `tcltk`) requires [XQuartz](#) to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your macOS to a new major version.

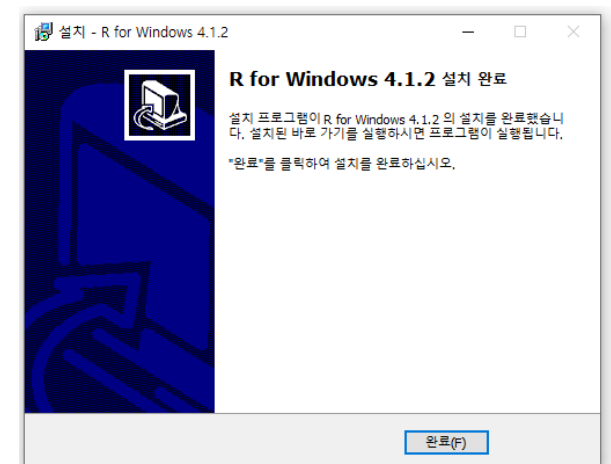
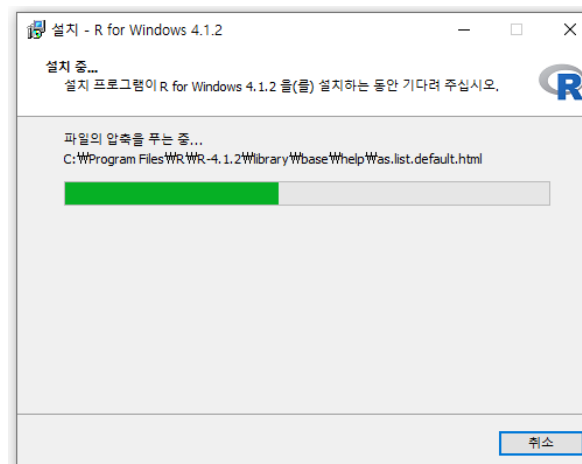
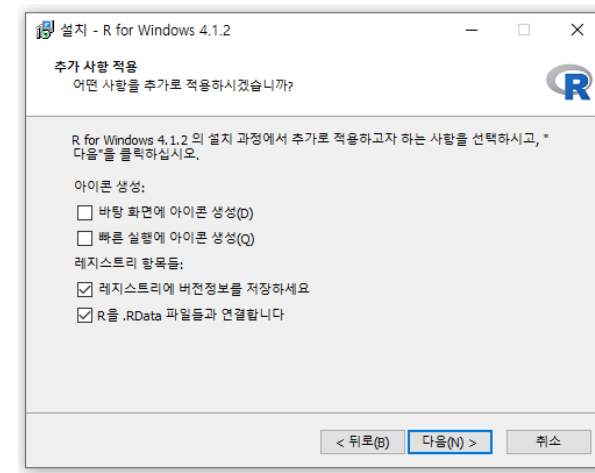
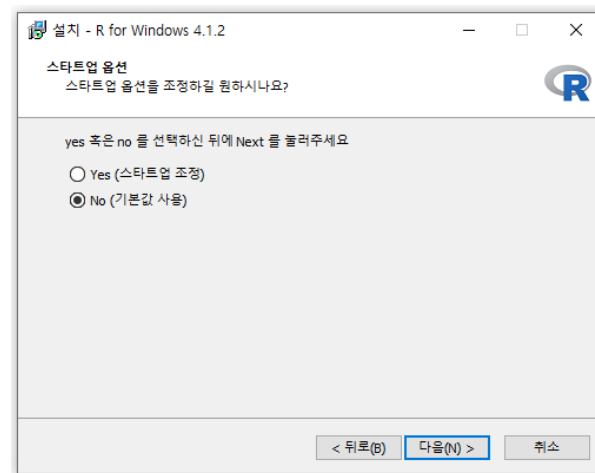
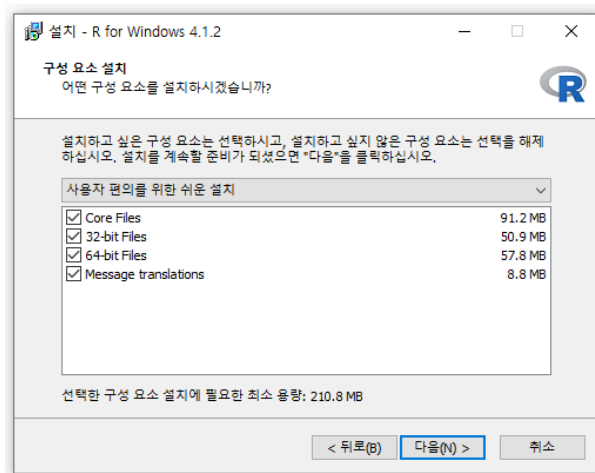
This release supports Intel Macs, but it is also known to work using Rosetta2 on M1-based Macs. For native Apple silicon arm64 binary see below.

Important: this release uses Xcode 12.4 and GNU Fortran 8.2. If you wish to compile R packages from sources, you may need to download GNU Fortran 8.2 - see the [tools](#) directory.

R 설치 과정

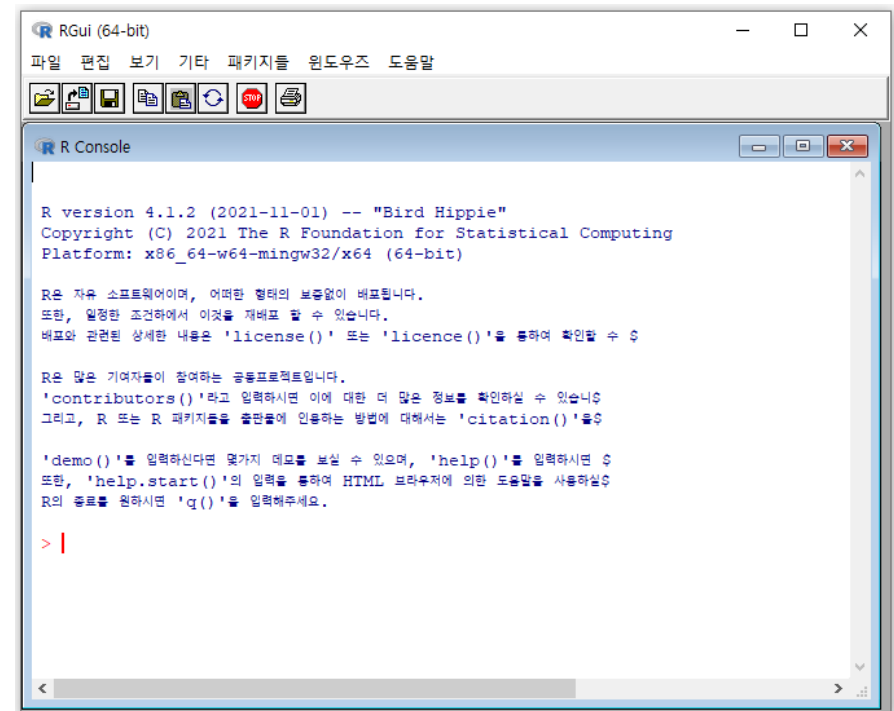
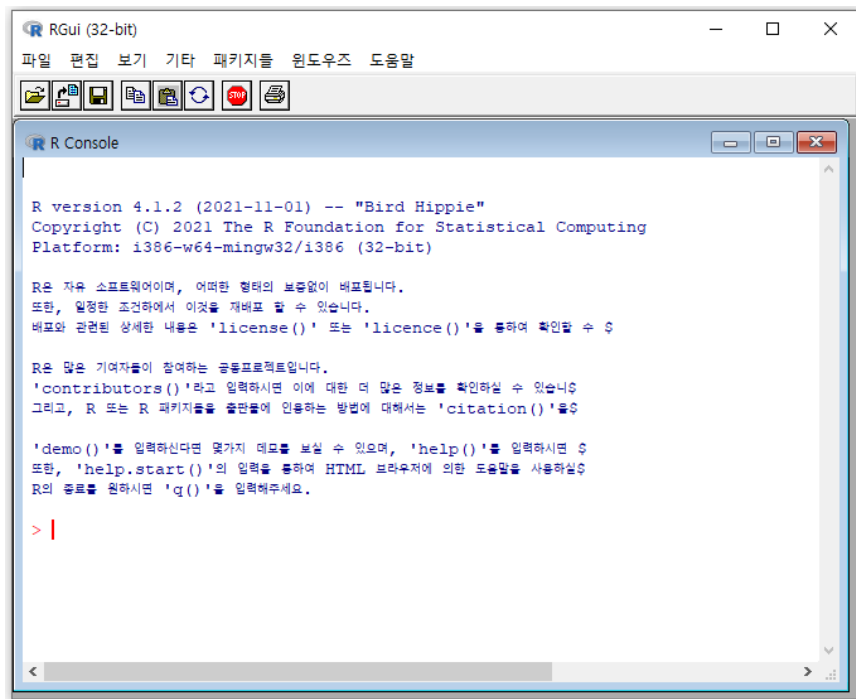
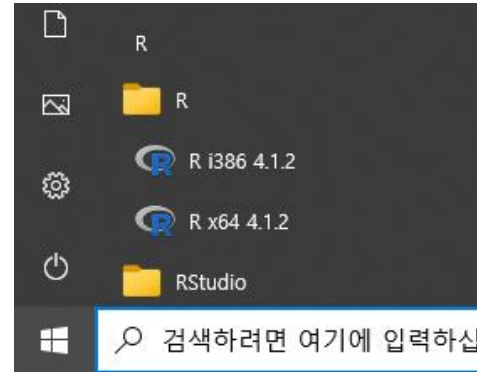
이름

R-4.1.2-win



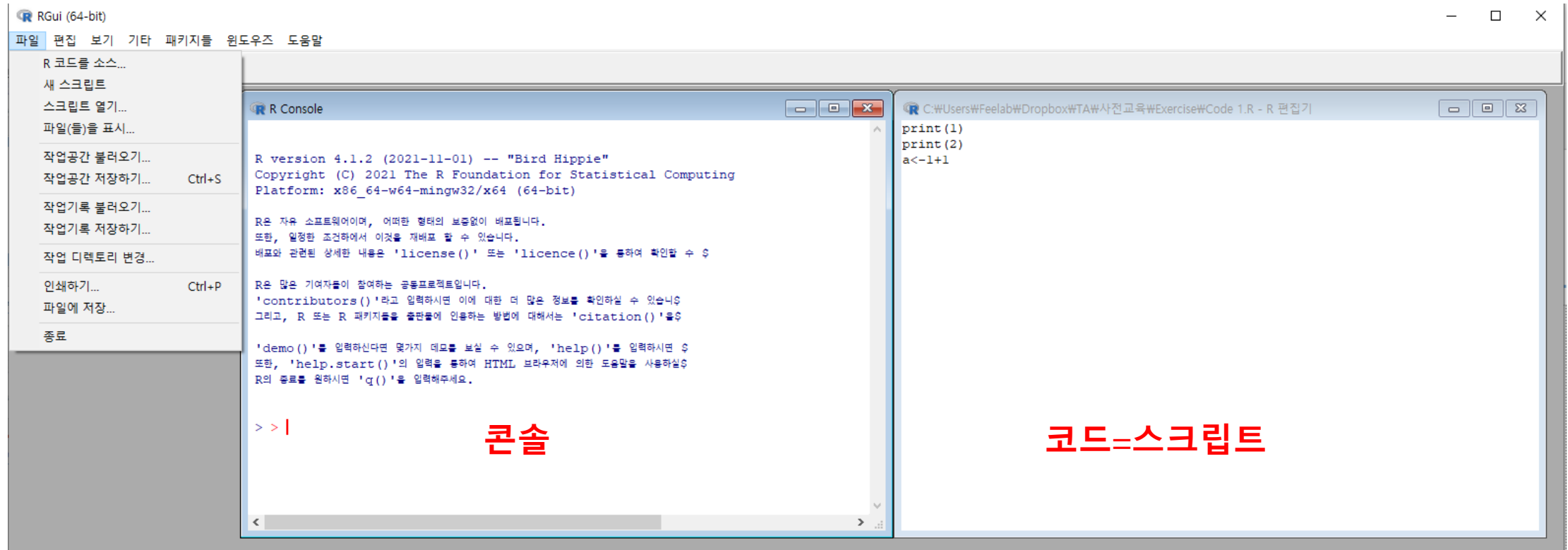
R 실행화면

- 시작버튼
- R
 - 32-bit: faster, lower RAM upto 3GB
 - **64-bit: no limited RAM: (Big Data)**



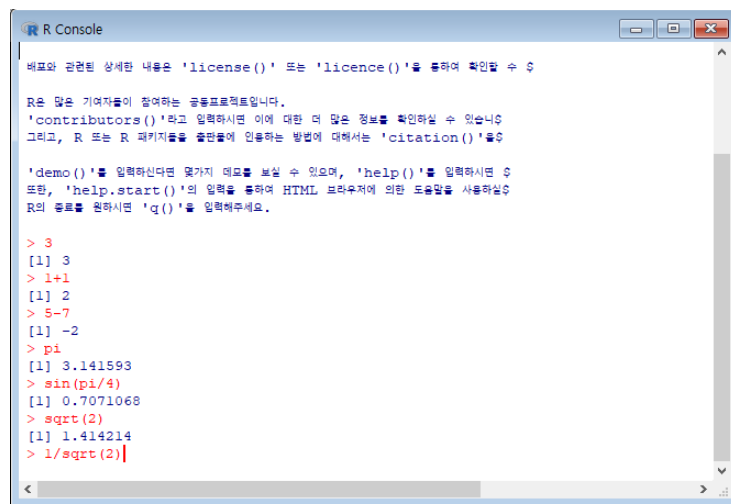
3. R 코딩 기초

R 프로그램의 용어와 구조 이해



R 프로그램의 용어와 구조 이해

- 콘솔
 - 입력을 기다리는 prompt
 - 코드를 복사에서 붙여넣기 하면 line-by-line 으로 실행됨
- [실습 1] 콘솔창 사용
 - 출력 해보기
 - `print(1)`
 - “Lecture Note R Code.R” 에 있는 “#Example 1” 의 코드 복사해서 붙여넣기



```
R Console

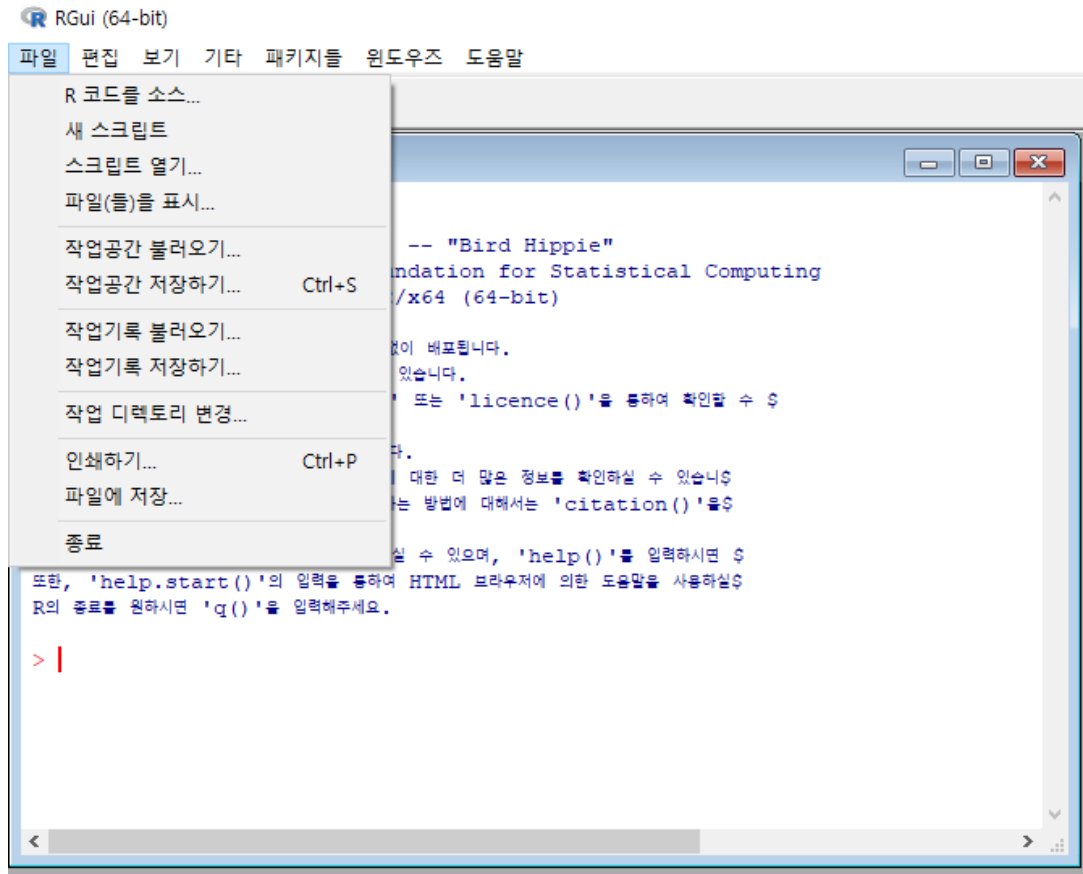
비표와 관련된 상세한 내용은 'license()' 또는 'licence()'를 통하여 확인할 수 있음

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()'을 입력하십시오

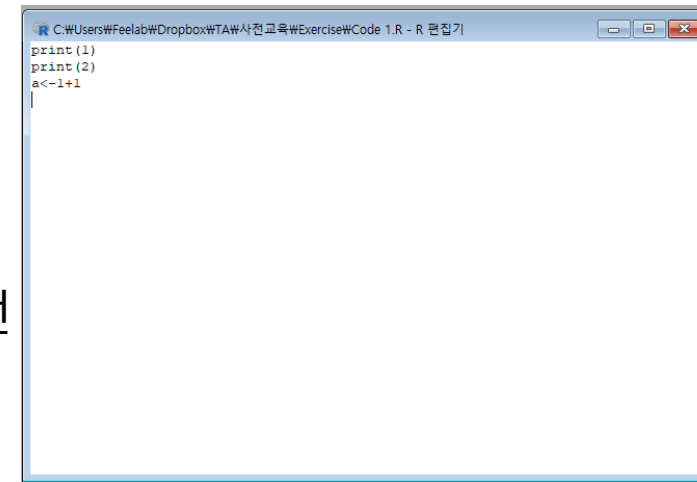
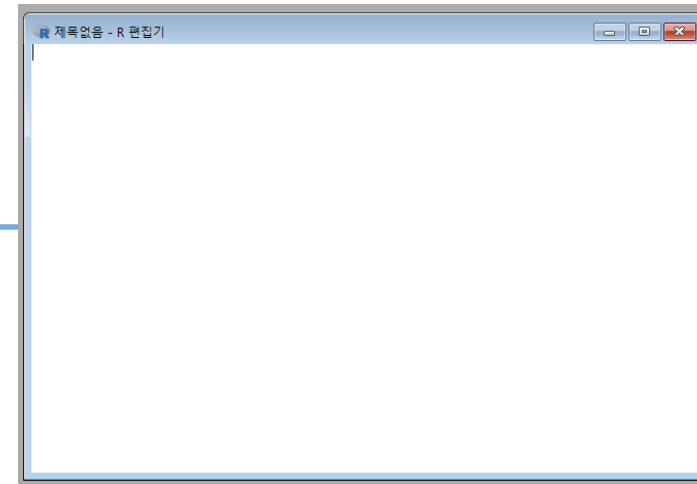
'demo()'를 입력하신다면 몇가지 데모를 보실 수 있으며, 'help()'를 입력하시면 도움말, 'help.start()'의 입력을 통하여 HTML 브라우저에 의한 도움말을 사용할 수 있습니다
R의 도움을 원하시면 'q()'를 입력하십시오.

> 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)
```

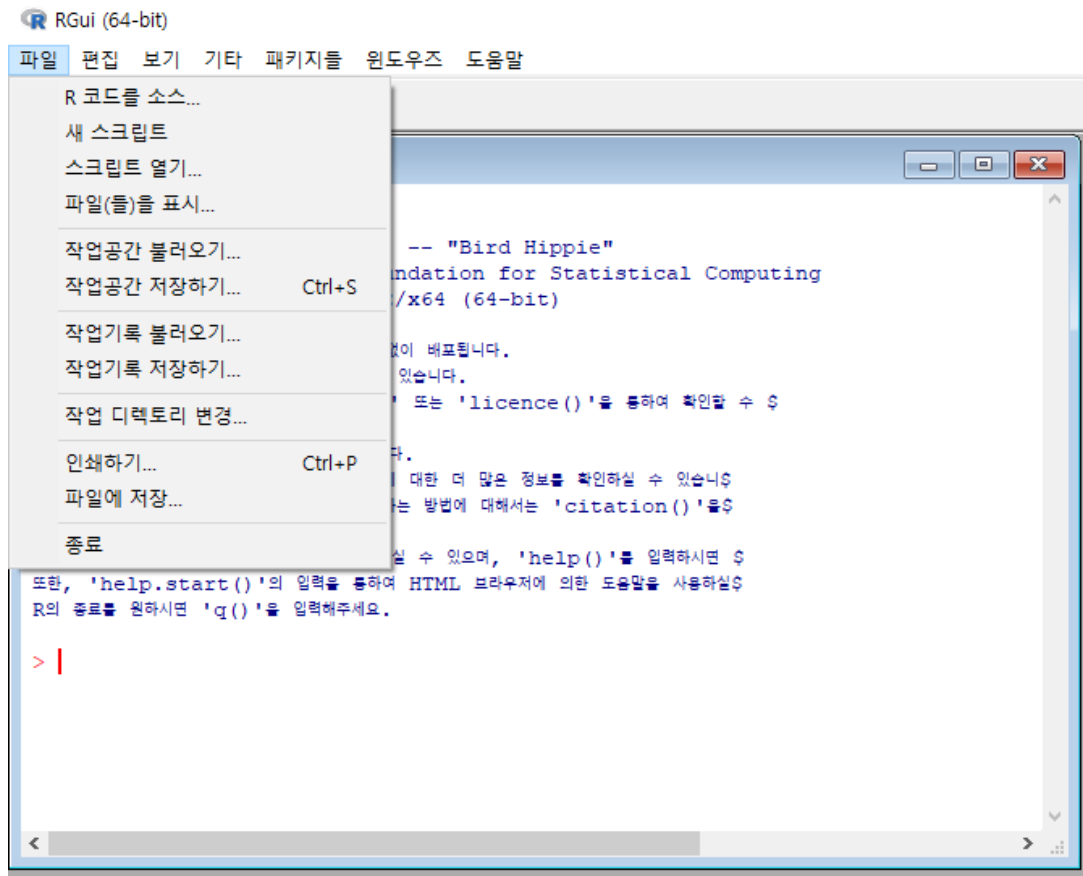
R 프로그램의 용어와 구조 이해



- 새 스크립트
 - 스크립트=코드
 - R 코드를 작성
 - 코드 작성 후 저장
- 스크립트 열기
 - 기존의 작성되었던 스크립트(코드) 열기



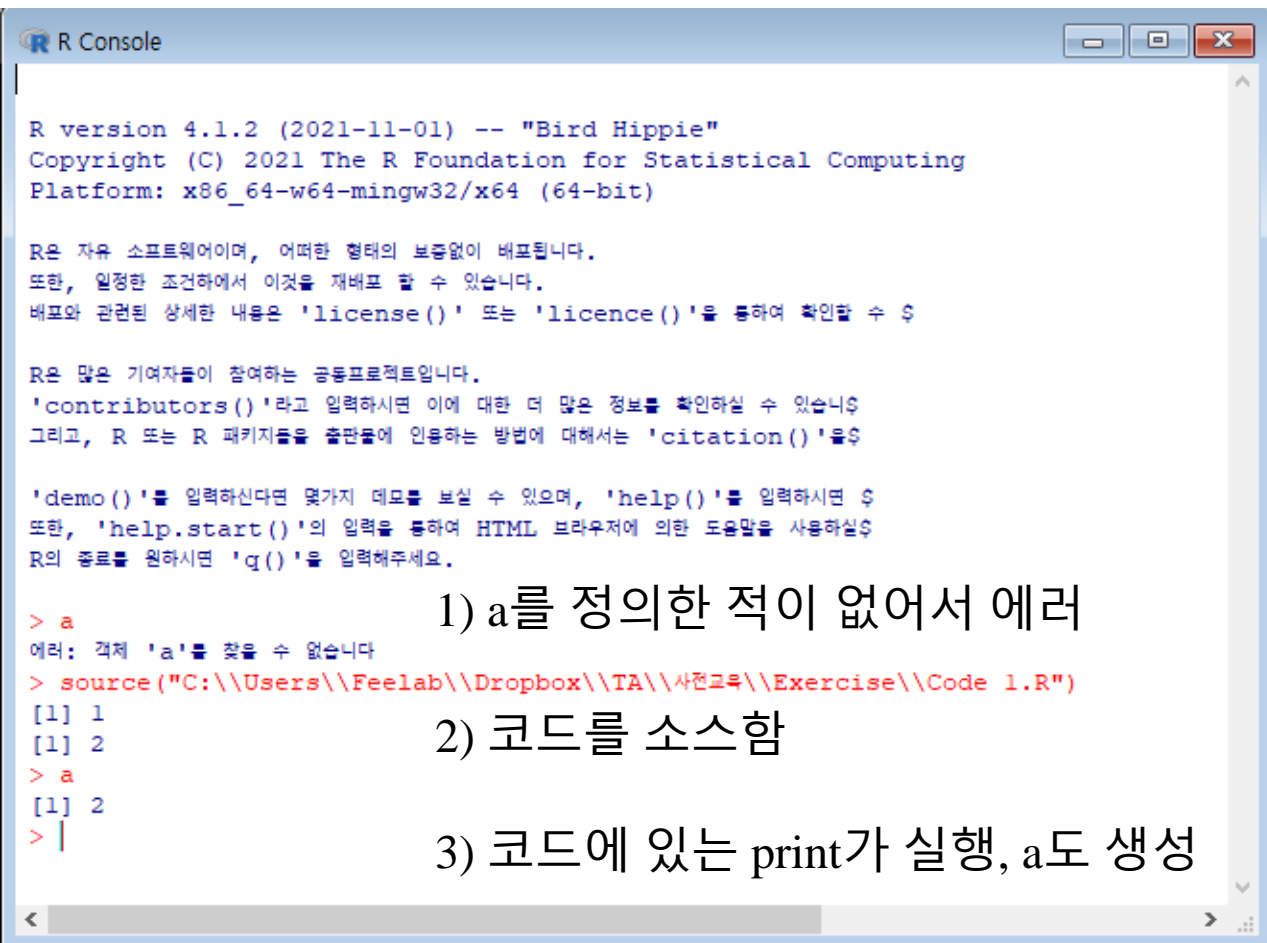
R 프로그램의 용어와 구조 이해



- R 코드를 소스....
 - 기존의 작성되었던 코드에 있는 정보를 읽어옴
 - 즉, source 도 함수임.

R 프로그램의 용어와 구조 이해

- R 코드를 소스....
 - 기존의 작성되었던 코드에 있는 정보를 읽어옴. 즉, source 도 함수임.



```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 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 패키지들을 출판물에 인용하는 방법에 대해서는 'citation()'을 보세요.

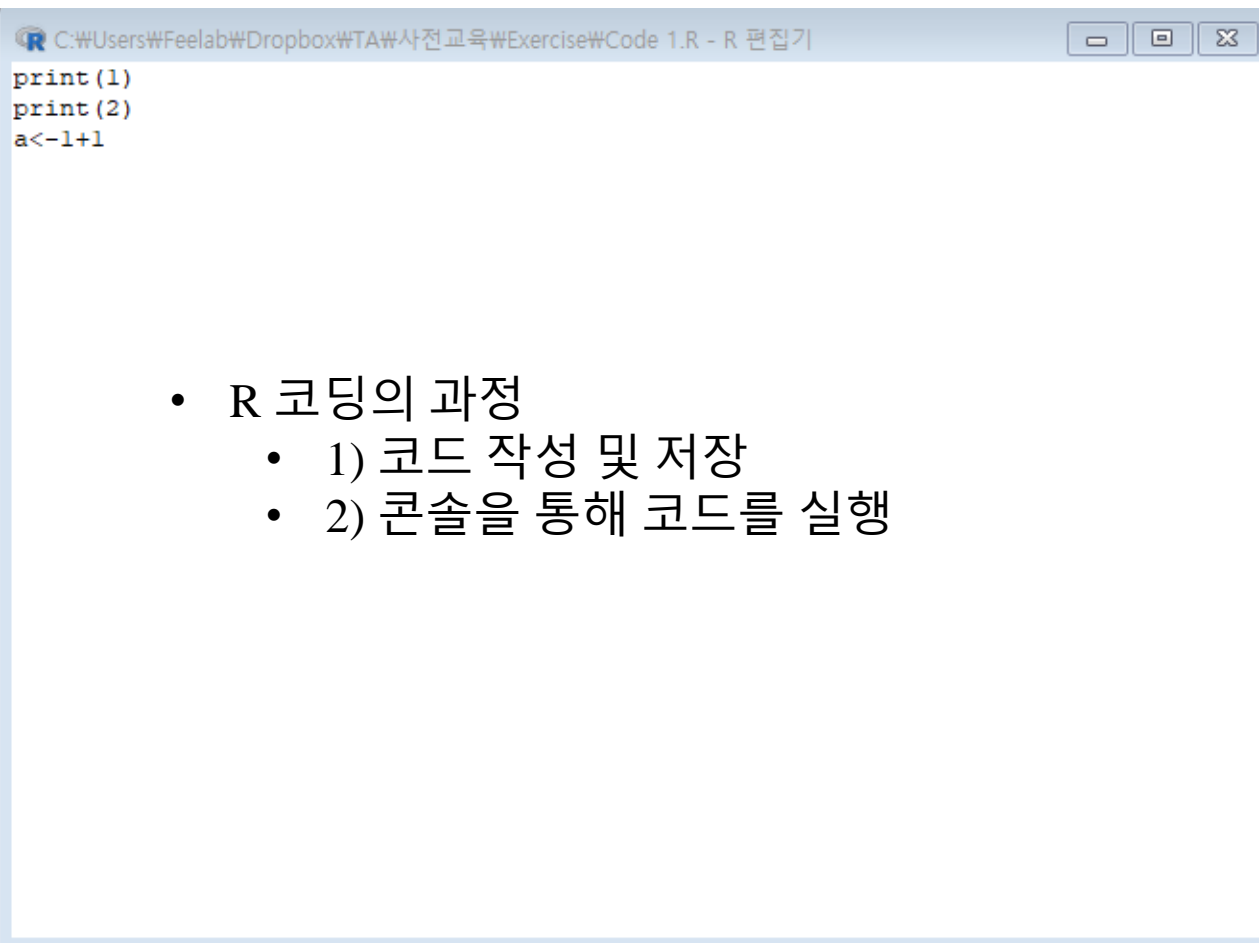
'demo()'를 입력하신다면 몇가지 데모를 보실 수 있으며, 'help()'를 입력하시면 R의 종류를 원하시면 'q()'를 입력해주세요.

> a
에러: 객체 'a'를 찾을 수 없습니다
> source("C:\\Users\\Feelab\\Dropbox\\TA\\사전교육\\Exercise\\Code 1.R")
[1] 1
[1] 2
> a
[1] 2
> |
```

1) a를 정의한 적이 없어서 에러

2) 코드를 소스함

3) 코드에 있는 print가 실행, a도 생성

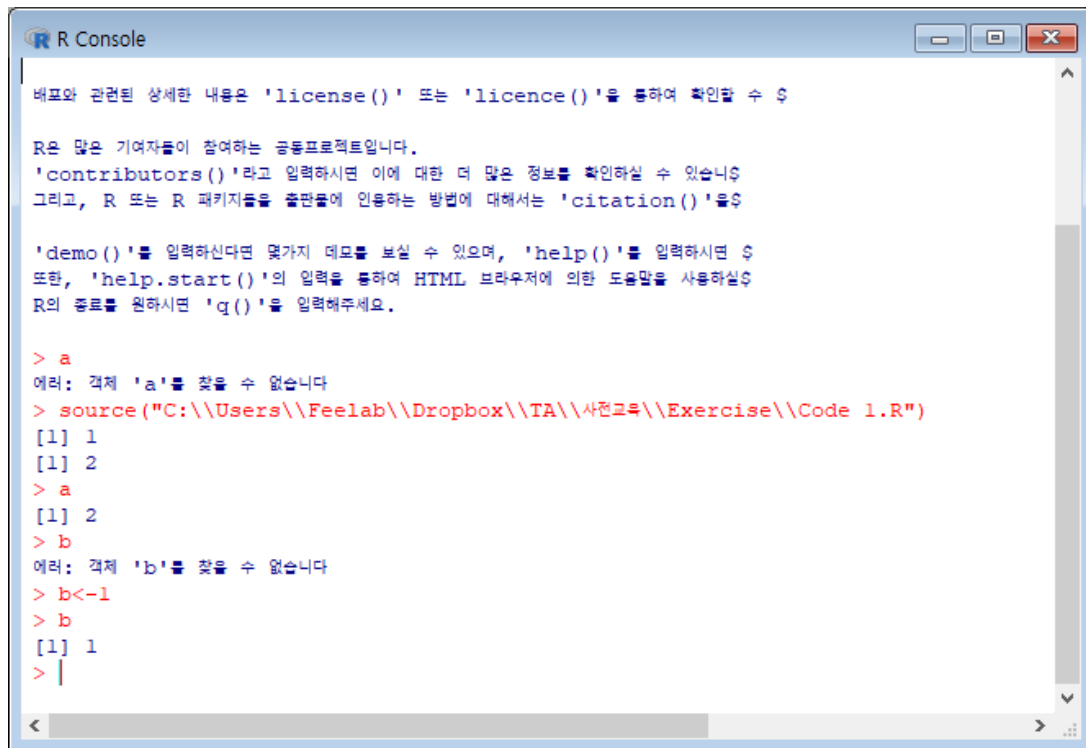


```
C:\\Users\\Feelab\\Dropbox\\TA\\사전교육\\Exercise\\Code 1.R - R 편집기

print(1)
print(2)
a<-1+1
```

- R 코딩의 과정
 - 1) 코드 작성 및 저장
 - 2) 콘솔을 통해 코드를 실행

R 프로그램의 용어와 구조 이해



```
R Console

배포와 관련된 상세한 내용은 'license()' 또는 'licence()'을 통하여 확인할 수 있습니다.

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

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

> a
에러: 객체 'a'를 찾을 수 없습니다
> source("C:\\Users\\Feelab\\Dropbox\\TA\\사전교과목\\Exercise\\Code 1.R")
[1] 1
[1] 2
> a
[1] 2
> b
에러: 객체 'b'를 찾을 수 없습니다
> b<-1
> b
[1] 1
> |
```

- R 콘솔 창을 활용하여 직접 코딩도 가능
 - b가 없어서 에러
 - b에 1을 대입
 - b<-1
 - b를 출력하면 1이 나옴.
- Console 창의 활용
 - 코드의 결과물을 확인
 - 절대로 콘솔 창에 직접 장문의 코딩은 하지 말기
 - 저장과 코드 관리가 어려움
 - R 코드로 작업 관리

R 프로그램의 용어와 구조 이해

- [실습 2] 코드 작성 후 저장
 - “Code 1.R” 이란 이름으로 해당 내용을 타이핑 후 저장. 확장자는 자동으로 지정됨.
 - `print(1)`
 - `print(2)`
 - `a<-1+1`
- [실습 3] 코드 불러오기
 - R을 닫고 다시 실행.
 - 콘솔에서 a 확인
 - “Code 1.R” 불러오기
 - 콘솔에서 a 확인
- [실습 4] 코드 source하기
 - “Code 1.R” source해보기
 - 콘솔에서 a 확인

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

- [실습 5] 연산자 결과 확인
 - “**Lecture Note R Code.R**” 에 있는 “**#Example 4**”에 있는 코드들 한줄씩 실행해보기
- 사칙연산과 거듭제곱은 아주 많이 활용
- 나눗셈 몫이나 나머지도 적재적소에 활용가능
 - 그림 7장을 3 by 3 그리드 위에 배열하기
 - 행 좌표: `i %/% 3`
 - 열 좌표: `i %% 3`

R 코딩 기초

- 기본 수학 함수

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

R 코딩 기초

- [실습 6] 연산자 결과 확인
 - “**Lecture Note R Code.R**” 에 있는 “**#Example 4**”에 있는 코드들 한줄씩 실행해보기
 - $\log(\exp(1))$ 의 값 확인하기
- 경제학, 금융, 사회과학 분야에서 삼각함수를 쓸 일은 많지는 않음
- 지수 로그는 아주 많이 활용
 - 지수함수: continuous 복리계산 등
 - 로그함수: 변수변환
 - Outlier효과를 경감시킴
 - 탄력성 (elasticity)로 **회귀분석결과를 해석가능하도록** 해줌.

R 코딩 기초

- Comment (주석)
 - 주석 : 코드 이해를 돕기 위해 덧붙이는 설명
 - 어려운 코드에 대한 보충 설명
 - 추후 작업을 위한 책갈피 역할
 - 코드 흐름에 영향을 주지 않음.
 - 반드시 # 이후에 입력. # 를 붙이지 않으면 Error 발생
- [실습 7] 주석의 역할 이해
 - 콘솔창에 x 입력
 - 에러 확인
 - #x 입력
 - 에러나는지 확인
- [실습 8] 코드내에서 주석의 역할 이해
 - 해당 코드 작성 후 “Code 2.R”로 저장
 - `x<-1`
 - `#y<-2`
 - 코드 소스하여 x만 생성되는지 확인

R 코딩 기초

- R 프로그램 종료 (Quit R)
 - 콘솔에 `quit()` 입력
 - 콘솔 내용 저장 시 [예], 그렇지 않으면 [아니요] 선택
- 변수는 언제 사라지는가?
 - `rm(list=ls())`
 - 콘솔을 초기화시키고 다시 처음부터 코딩하는 습관들이기
 - 콘솔이 초기화되지 않으면 다른 협업자가 코드를 받았을 땐 선언하기 전 변수를 사용하게 되어 에러가 발생할수도!
 - 과제나 시험 채점 시 감점 대상!

4. 데이터 종류

데이터 종류

자료 형태	구성 차원	자료 유형	복수 데이터 유형 적용 여부
스칼라(Scalar)	0차원	수치/문자/복소수/논리	불가능
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: 스칼라 (Scalar)

- 스칼라 (Scalar) : 특정 형태의 자료
- `x <- 1`
- `y <- 2`
- `z <- TRUE`
- `w <- "w"`

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] 1.57312457 -0.47063194 -1.05106000 -0.49465712 -0.26027291 0.65970928 0.22122471
[8] -0.28235983 -0.84233511 1.38652094 0.37525593 0.22668169 0.35378657 -2.60994569
[15] -1.23454556 0.68156419 1.38090300 -1.38234872 1.10226228 0.83901573 -0.46790396
[22] 1.96601037 2.06990312 1.00173965 1.22941515 0.06399120 1.92895784 -0.06726226
[29] 0.39344149 1.11172415
```

- 벡터의 예
 - 삼성전자 주가
 - 삼성전자 실적

데이터 종류: 벡터 (Vector)

- 벡터와 관련된 연산

- Summary functions

- length(x)

- mean(x)

- sd(x)

- range(x)

- summary(x)

- 정렬

- sort(x)

- sort(x, decreasing = TRUE)

```
> length(x)
[1] 5
> mean(x)
[1] 3
> sd(x)
[1] 1.581139
> range(x)
[1] 1 5
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
     1      2       3       3       4       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 (인덱싱, indexing, 매우 중요!)
 - 데이터 다루기는 결국 인덱싱의 문제이다.
- 인덱싱 기본 문법
 - `[]` 를 활용하여 각 원소에 접근
 - `x[1]` : 첫 번째 원소
 - `x[2]` : 두 번째 원소
 - `x[-2]` : 두 번째를 제외한 나머지 원소들
- 인덱싱을 활용한 데이터 변환
 - `x[3] <- 4` : 세 번째 원소(5)를 4로 바꿈
- 고급 인덱싱: 조건부 (conditional) 인덱싱
 - `x[2<x&x<5]` : $2 < x < 5$ 인 원소 모두 출력

```
> 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] # x의 세 번째 원소 확인
[1] 5
> x[3] <- 4 # x의 세 번째 원소(5)를 4로 바꿈
> x # x 확인
[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: 벡터 (Vector)

- [실습 9] Conditional indexing (조건부 참조)
 - & 연산자: and
 - | 연산자: or
 - <= : 이하
 - >= : 이상
 - | 연산자를 활용하여 x에서 2보다 작거나 같거나 5보다 크거나 같은 원소들을 불러오기
 - `x[x<=2|x>=5]`
- [실습10] Conditional indexing (조건부 참조)
 - == : equal
 - | 연산자를 활용하여 x에서 2보다 작거나 같거나 5보다 크거나 같은 원소 또는 3인 원소 불러오기
 - `x[(x<=2|x>=5) | (x==3)]`

데이터 종류: 벡터 (Vector)

- Vector indexing의 원리 이해
 - `x <- c(1,2,3,4,5)`
 - Index로는 **scalar**나 **vector**가 가능하다.
 - `x[0]`
 - `x[c(1,2)]`
 - `x[c(4,5)]`
- 그렇다면 conditional indexing을 하는 원리는?
 - `x<5`
`> x<5`
`[1] TRUE TRUE TRUE TRUE FALSE`
 - 즉, `x<5` 도 벡터로서 저장이 된다.
 - 따라서 `x<5`를 벡터로 인식하여 인덱싱에 사용할 수 있다.

데이터 종류: 벡터 (Vector)

- 복습

- 데이터 타입 vector에는 다른 종류의 원소도 저장될 수 있다?

- Y/N

```
> x<-c(1, "w")
> x
[1] "1" "w"
> x[0]
character(0)
> x+1
x + 1에서 다음과 같은 에러가 발생했습니다:이항연산자에 수치가 아닌 인수입니다
> x <- c(TRUE, "Apple")
> x
[1] "TRUE" "Apple"
> x[0]
character(0)
> x[0]==TRUE
logical(0)
```

- Vector는 인덱싱에 사용할 수 없다?

- Y/N

데이터 종류: 행렬 (Matrix)

- 행렬(Matrix) : 동일한 형태의 자료를 2차원 형태로 여러 개 모아서 취급하는 데이터 형태
 - 가장 널리 활용되는 자료형태 중 하나
 - 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

- 1) 벡터의 생성

```
> 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
> vec1
[1] 1 2 3
> vec2
[1] 4 5 6
> vec3
[1] 7 8 9
```

- 2) 벡터들 합치기

- rbind: row 방향으로 bind한다.

```
> mat1 <- rbind(vec1, vec2, vec3)
> mat1
      [,1] [,2] [,3]
vec1     1     2     3
vec2     4     5     6
vec3     7     8     9
```

- cbind: column 방향으로 bind한다.

```
> mat2 <- cbind(vec1, vec2, vec3)
> mat2
      vec1 vec2 vec3
[1,]     1     4     7
[2,]     2     5     8
[3,]     3     6     9
```

데이터 종류: 행렬 (Matrix)

- 만약 다른 데이터 형태의 vector를 합친다면?

```
> vec4 <- c("a","b","c") # make vector ("a","b","c")
> mat1b <- rbind(vec1, vec2, vec4) # row (행) 방향으로 행렬 생성
> mat1b
      [,1] [,2] [,3]
vec1 "1"  "2"  "3"
vec2 "4"  "5"  "6"
vec4 "a"  "b"  "c"
```

- Elements are not number!

데이터 종류: 행렬 (Matrix)

- 행렬(Matrix) 생성
 - 함수 “matrix” 와 parameter인 “nrow”를 사용
- R 함수 참고
 - 함수의 argument가 궁금할 경우 오른쪽과 같이 documentation을 검색할수도
 - <https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/matrix>

RDocumentation

Search all packages and functions

base (version 3.6.2)

matrix: Matrices

Description

``matrix`` creates a matrix from the given set of values.

``as.matrix`` attempts to turn its argument into a matrix.

``is.matrix`` tests if its argument is a (strict) matrix.

Usage

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
       dimnames = NULL)

as.matrix(x, ...)
# S3 method for data.frame
as.matrix(x, rownames.force = NA, ...)

is.matrix(x)
```

데이터 종류: 행렬 (Matrix)

- 행렬(Matrix) 생성
 - 함수 “matrix” 와 parameter인 “nrow”를 사용

```
> mat3 <- matrix(1:12, nrow = 3)
> mat3
```

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

```
> mat3b <- matrix(1:12, ncol = 4)
> mat3b
```

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

- [실습 11] Matrix 함수 익히기
 - Matrix함수의 argument를 잘 참고하여 어떻게 하면 아래와 같은 행렬을 만들 수 있을까?

```
> mat3c
```

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

- mat3c <- matrix(1:12, ncol = 4, byrow=TRUE)

데이터 종류: 행렬 (Matrix)

- Column 이름 생성
 - byrow=FALSE
 - dimnames에 행, 열의 이름 지정
- byrow=TRUE

```
> 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
```

```
> mat4b <- matrix(1:12,  
+                nrow = 4,  
+                dimnames = list(c("C1", "C2", "C3", "C4"),  
+                                c("R1", "R2", "R3")),  
+                byrow=TRUE  
+                ) # 행과 열 이름 설정
```

```
> mat4b  
   R1 R2 R3  
C1  1  2  3  
C2  4  5  6  
C3  7  8  9  
C4 10 11 12
```

데이터 종류: 행렬 (Matrix)

- Element
 - `mat1[1]`
 - `mat1[1,1]`
 - `mat1[1,2]`
 - `mat1[1,2,drop=FALSE]`
 - # return the element as matrix
- Row
 - `mat1[1,]`
 - # return row vector
 - `mat1[1,,drop=FALSE]`
 - # return row matrix
- [실습 12] Column
 - `mat1`의 첫번째 column을 가져오려면?
 - `mat1[,1]`
 - `mat1`의 두번째 column을 가져오려면?
 - `mat1[,2]`
 - `mat1`의 두번째 column을 matrix 형태로 가져오려면?
 - `mat1[,2,drop=FALSE]`

데이터 종류: 행렬 (Matrix)

- Submatrix

- `mat1[1:2, 2:3]`

- 1:2의 정체는?

```
> 1:2  
[1] 1 2
```

- `mat1[1:1,2:3]`

- `mat1[1:1,2:3,drop=FALSE]`

- `c()`를 이용하여 submatrix 가져오기

- `mat1[c(1)]`

- `mat1[c(1),]`

- `mat1[c(1,3)]`

- `mat1[c(1,3),]`

- [실습 13] 첫번째 행, 세번째 행에 있는 첫번째 열, 세번째 열을 가져오려면?

- `mat1[c(1,3),c(1,3)]`

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
> chars
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

- 단어도 저장 가능

```
> chars2
[1] "apple"      "banana"    "card"      "dimension"  "erorr"
[6] "fire"       "good"      "high"      "identifiers" "james"
```

데이터 종류: 행렬 (Matrix)

- Reshaping

- mat6

```
> mat7 <- matrix(1:24, nrow=8)
> mat7
      [,1] [,2] [,3]
[1,]    1    9   17
[2,]    2   10   18
[3,]    3   11   19
[4,]    4   12   20
[5,]    5   13   21
[6,]    6   14   22
[7,]    7   15   23
[8,]    8   16   24
```

- 만약 2 by 12 matrix로 만들고 싶다면?

- But... why?

- 만약 mat7이 각 분기의 observation을 한 row마다 나열한 것이라고 하자.
 - 어떤 필요에 의해서 1년치의 observation으로 변환해야할 수 있다.
 - 예) stock return momentum 계산
 - 과거 12개월 치 return으로

- dim(mat7) <- c(2,12)

데이터 종류: 행렬 (Matrix)

- 행렬(Matrix) 연산
 - R의 행렬연산은 행렬 곱이 아님!
- Multiply scalar
 - $x1 * 3$
- Multiply vector
 - $x1 * c(10, 20)$
- [실습 14] 그렇다면 다음 계산의 결과를 생각해 보자
 - $x1 * c(10, 20, 30, 40)$
- [실습 15] 각각 Column에 10, 20, 30, 40을 곱하려면?
 - $x1 * rep(c(10, 20, 30, 40), 2)$

데이터 종류: 행렬 (Matrix)

- 행렬(Matrix) 연산
 - apply: 행 또는 열에 적용할 수 있는 함수 (Max, Min, Mean, Sum 등)

RDocumentation

Search all packages and functions

base (version 3.6.2)

apply: Apply Functions Over Array Margins

Description

Returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

Usage

```
apply(X, MARGIN, FUN, ...)
```

Arguments

X	an array, including a matrix.
MARGIN	a vector giving the subscripts which the function will be applied over. E.g., for a matrix <code>`1`</code> indicates rows, <code>`2`</code> indicates columns, <code>`c(1, 2)`</code> indicates rows and columns. Where <code>`x`</code> has named dimnames, it can be a character vector selecting dimension names.
FUN	the function to be applied: see 'Details'. In the case of functions like <code>`+`</code> , <code>`%*%`</code> , etc., the function name must be backquoted or quoted.
...	optional arguments to <code>`FUN`</code> .

데이터 종류: 행렬 (Matrix)

- apply 함수
 - mat1
 - apply(mat1, 1, max)
 - 1: 각 행에 대해서 max 함수 적용한 결과값 반환
 - apply(mat1, 2, mean)
 - 2: 각 열에 대해서 mean 함수 적용한 결과값 반환
- [실습 16] 두번째 열에 c(1,2)를 대입해보기

데이터 종류: 행렬 (Matrix)

- 행렬(Matrix) 연산
 - Element-wise 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
```

- 행렬 곱

```
> x1 %*% x3 # 행렬의 곱셈
      [,1] [,2]
[1,]   178   242
[2,]   220   300
```

실전에서는?

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))`
 - `y <- array(1:120, dim=c(2,4,3,5))`
- 배열로 저장되어야 하는 데이터 종류는?
 - 이미지
 - 픽셀별 (x, y 좌표)별 RGB
 - 딥러닝 학습을 위한 시계열 데이터
 - 관측치 1: 다변수시계열
 - 예) 2000년~2010년까지의 주가수익률, lagged 시가총액, PER, PBR 등 (matrix 형태)
 - 관측치 2: 마찬가지로 다변수시계열
 - 2001년 ~ 2011년까지의 주가수익률, lagged 시가총액, PER, PBR 등 (matrix 형태)
- 이런 데이터의 경우 array 형태로 저장되어야하는 유형이다!

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) indexing

```
> x[1,,] # 1번째 row만을 모음
```

```
      [,1] [,2] [,3]  
[1,]    1    9   17  
[2,]    3   11   19  
[3,]    5   13   21  
[4,]    7   15   23
```

	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)

	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) indexing

```
> x[1,3,] # 1번째 row & 3번째 column만을 모음  
[1] 5 13 21
```

1	2	3
5	13	21

1번째 row, 3번째 column
에 속한 3개 원소들
Ex) 5 : (1,3,1) → (1)
Ex) 21 : (1,3,3) → (3)

	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)

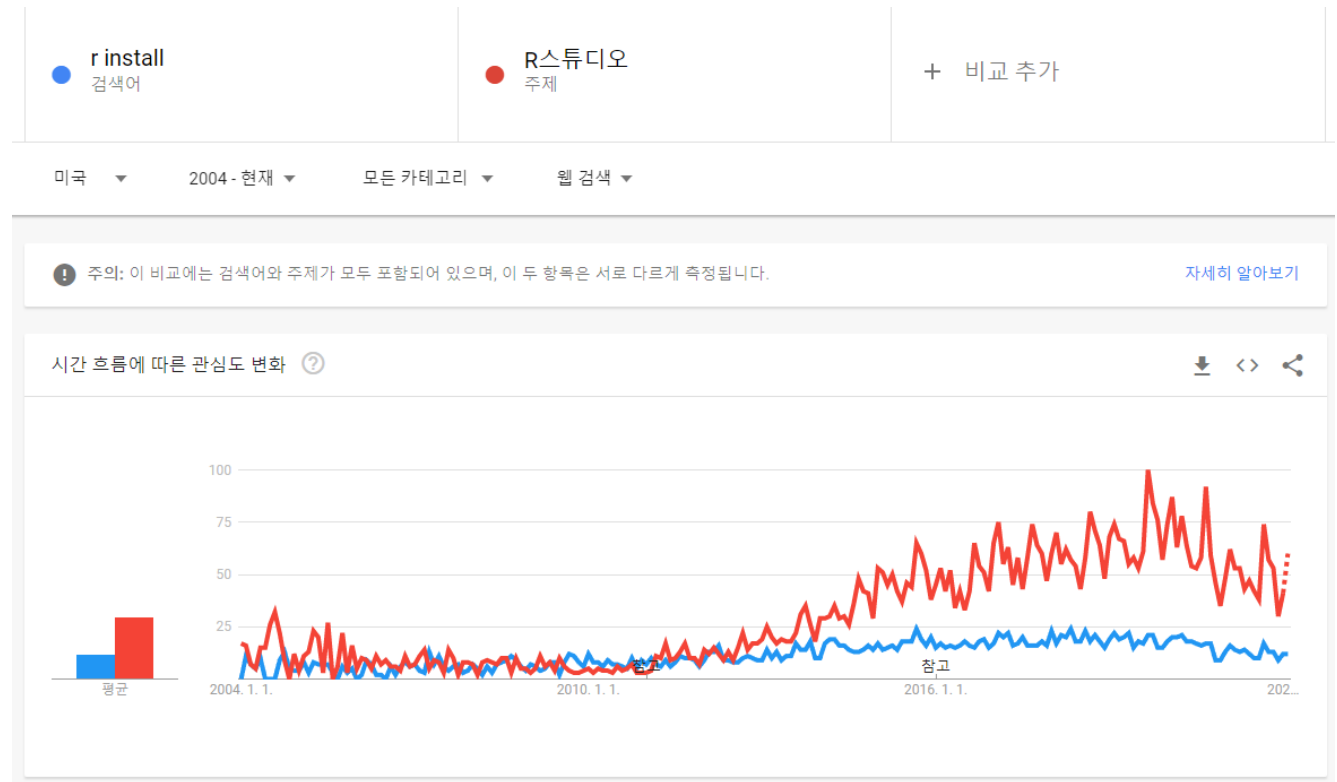
- [실습 16] Array indexing 결과 예상
 - 다음 코드를 실행한 결과가 무엇일지 생각해봅시다.
 - `x <- array(1:24, dim=c(2,4,3))`
 - `x[,2,]`
 - `x[,2,3]`

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

5. 실전 R

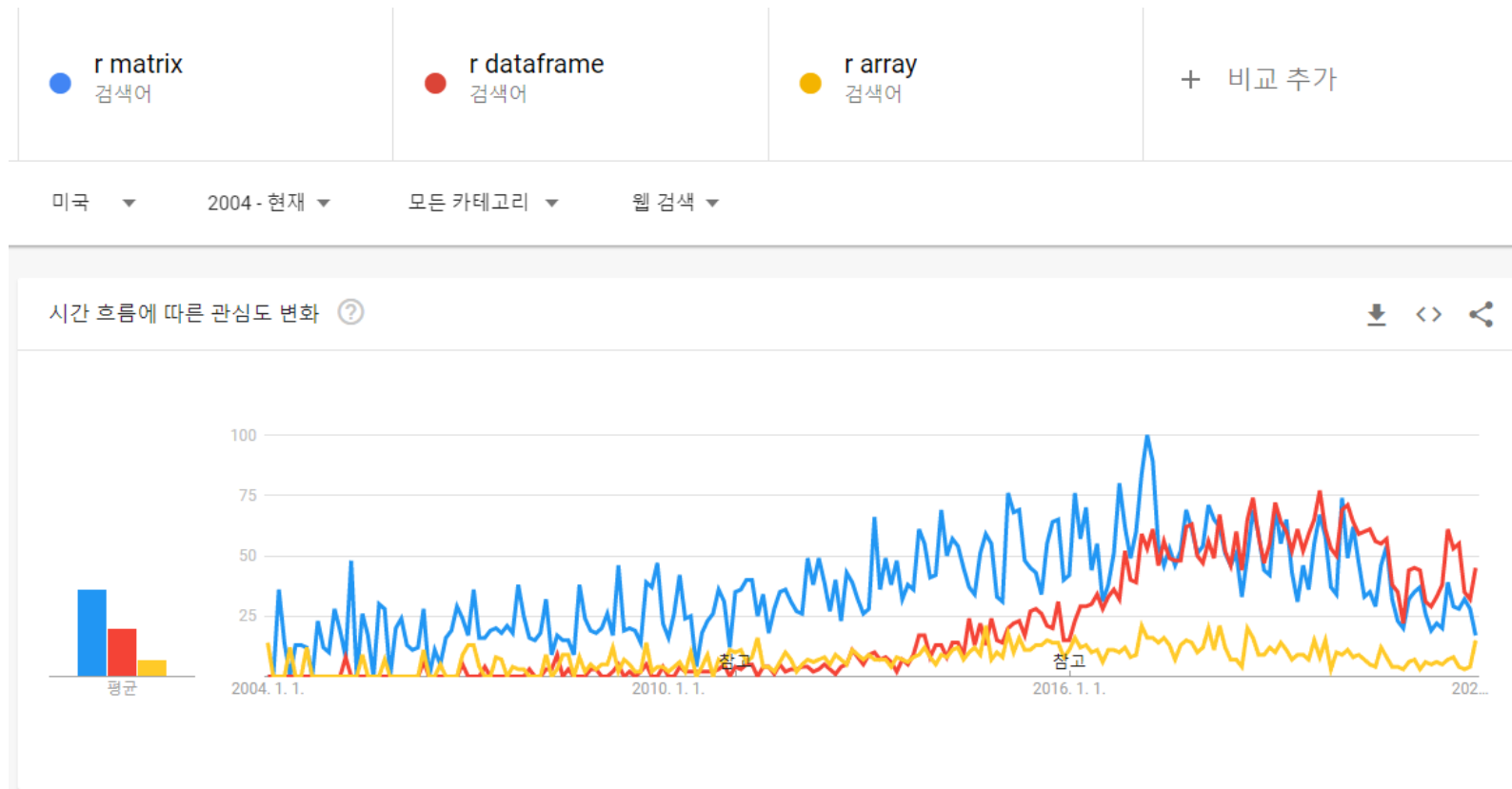
실전 R

- 수많은 변칙적 상황과 예상치 못한 오류와의 싸움
 - 수많은 데이터 타입
 - Outliers
 - (나오지 않는 결과)
 - (결과 나올 때까지 무한반복)
- 승률을 조금이라도 올리려면!
 - 도구를 잘 써야 한다.
 - Rstudio

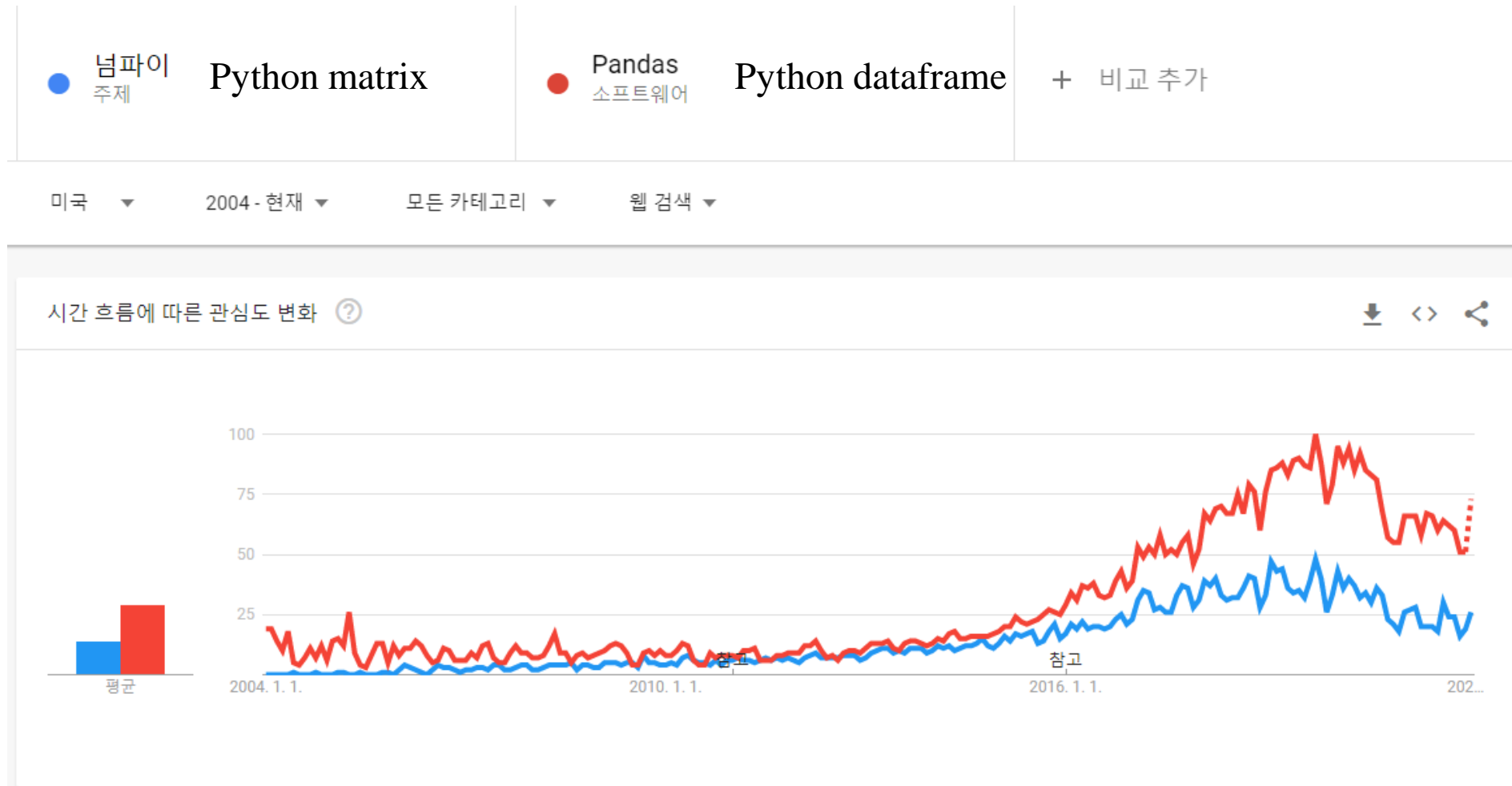


실전 R

- matrix vs data.frame (vs array) ?



실전 R (python)



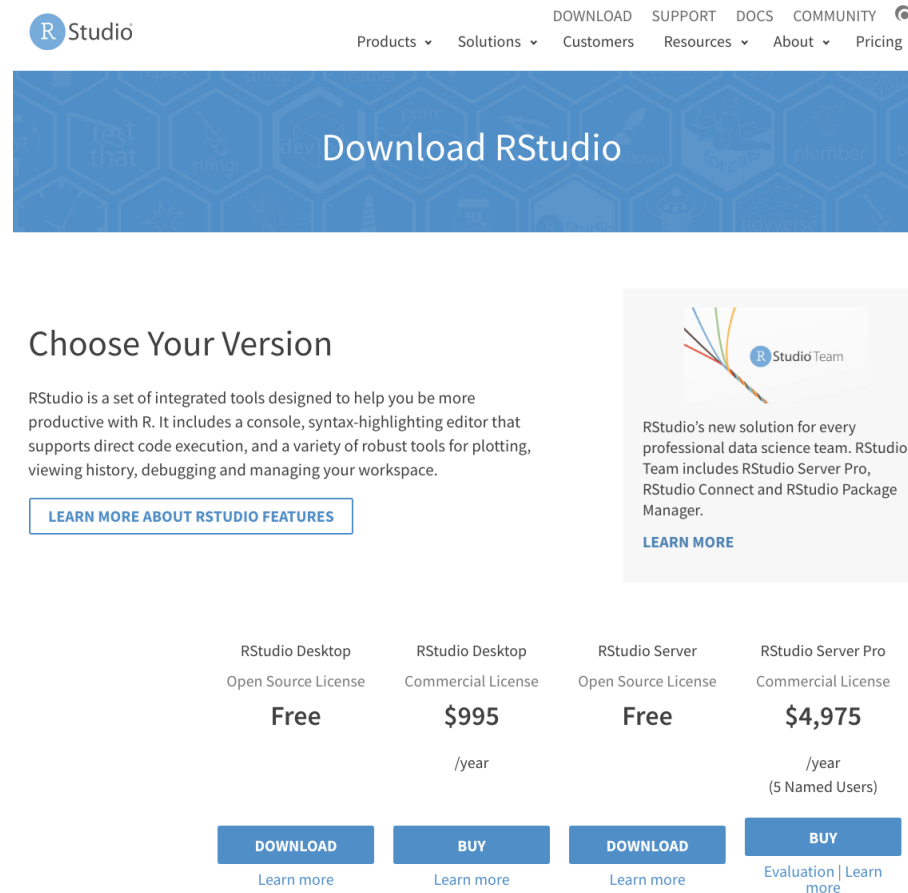
6. R studio 설치

R studio 설치

- R : 상호 대화식 인터페이스 (Python Jupyter Notebook과 비슷)
 - 입력된 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
 - IDE (Integrated Development Environment)

R studio 설치

- <https://rstudio.com/products/rstudio/download/>
 - [Rstudio Desktop] - [DOWNLOAD]



The screenshot shows the RStudio website's download page. At the top, there's a navigation bar with the RStudio logo and links for Products, Solutions, Customers, Resources, About, and Pricing. Below this is a large blue banner with the text "Download RStudio". Underneath the banner, there's a section titled "Choose Your Version" with a description of RStudio as a set of integrated tools. To the right of this section is a box for "RStudio Team" with a description of their new solution for professional data science teams. At the bottom, there's a table comparing four RStudio products: RStudio Desktop (Open Source License, Free), RStudio Desktop (Commercial License, \$995/year), RStudio Server (Open Source License, Free), and RStudio Server Pro (Commercial License, \$4,975/year for 5 named users). Each product has a "DOWNLOAD" or "BUY" button and a "Learn more" link.

RStudio Desktop	RStudio Desktop	RStudio Server	RStudio Server Pro
Open Source License	Commercial License	Open Source License	Commercial License
Free	\$995	Free	\$4,975
	/year		/year (5 Named Users)
DOWNLOAD	BUY	DOWNLOAD	BUY
Learn more	Learn more	Learn more	Evaluation Learn more

R studio 설치

- 운영체제 선택하여 설치

RStudio Desktop 1.3.1056 - Release Notes

1. Install R. RStudio requires R 3.0.1+.
2. Download RStudio Desktop. Recommended for your system:



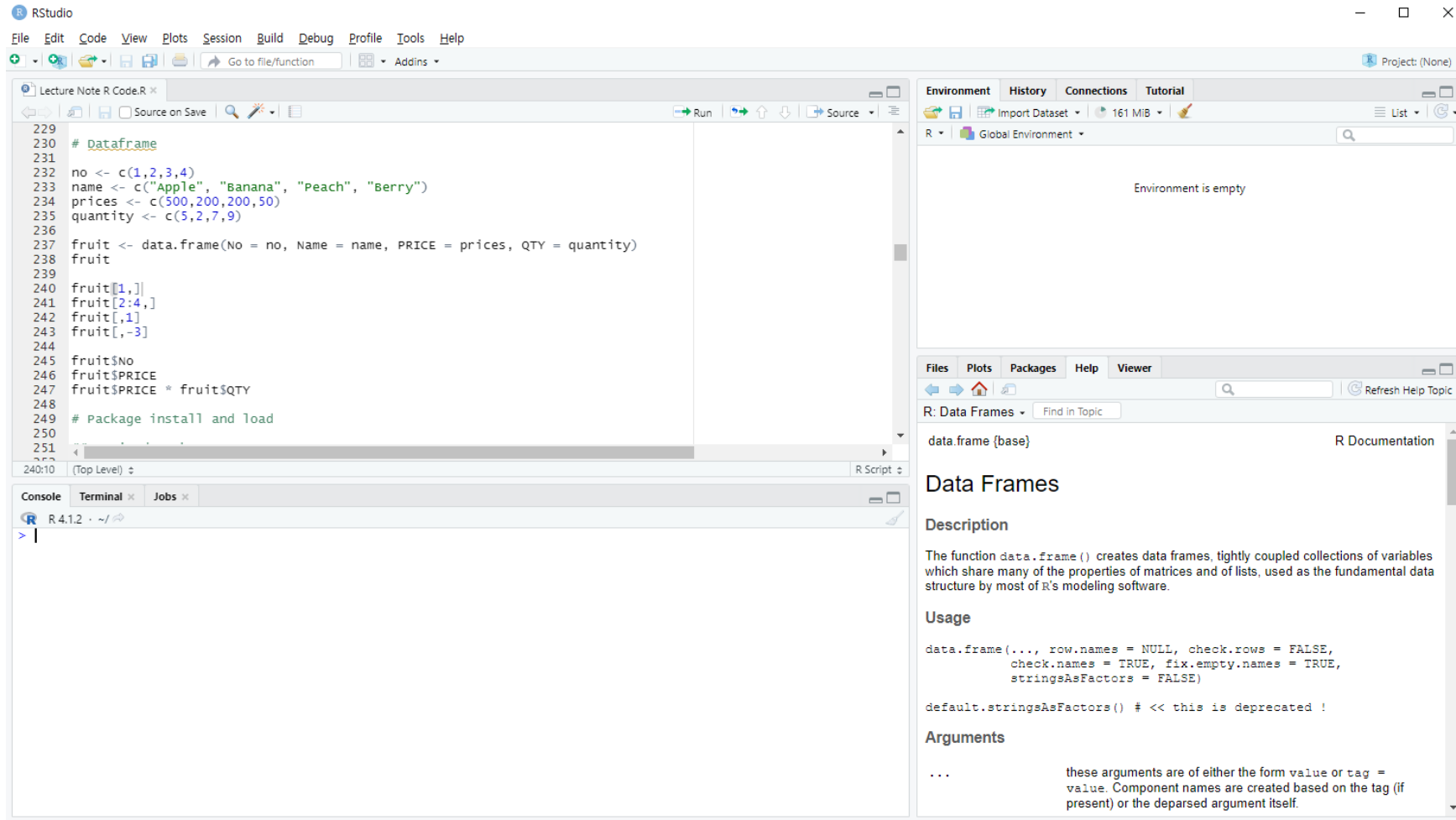
All Installers

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	a8f1fee5
macOS 10.13+	RStudio-1.3.1056.dmg	148.64 MB	f343c77d
Ubuntu 16	rstudio-1.3.1056-amd64.deb	124.56 MB	cbd5e5e5
Ubuntu 18/Debian 10	rstudio-1.3.1056-amd64.deb	126.50 MB	cd1a9e17
Fedora 19/Red Hat 7	rstudio-1.3.1056-x86_64.rpm	146.86 MB	0b1576bb
Fedora 28/Red Hat 8	rstudio-1.3.1056-x86_64.rpm	150.95 MB	bc4b3f44
Debian 9	rstudio-1.3.1056-amd64.deb	126.65 MB	3fb317e5
SLES/OpenSUSE 12	rstudio-1.3.1056-x86_64.rpm	119.17 MB	1be3540b
OpenSUSE 15	rstudio-1.3.1056-x86_64.rpm	128.14 MB	0e881257

R studio 둘러보기



R studio 둘러보기

The screenshot displays the RStudio environment with three main panels highlighted by red boxes:

- Source Editor (Top Left):** Contains R code for creating a data frame and performing calculations.


```

229 # Dataframe
230
231 no <- c(1,2,3,4)
232 name <- c("Apple", "Banana", "Peach", "Berry")
233 prices <- c(500,200,200,50)
234 quantity <- c(5,2,7,9)
235
236 fruit <- data.frame(No = no, Name = name, PRICE = prices, QTY = quantity)
237 fruit
238
239 fruit[[1,]]
240 fruit[2:4,]
241 fruit[,1]
242 fruit[,-3]
243
244 fruit$No
245 fruit$PRICE
246 fruit$PRICE * fruit$QTY
247
248 # Package install and load
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```
- Environment Panel (Top Right):** Shows the current environment with variables:

Variable	Class	Value
z	int [1:4, 1:6]	1 2 3 4 5 6 7 8 9 10 ...
w	int [1:2, 1:3, 1:4]	1 2 3 4 5 6 7 8 9 10 ...
x	1	1
y	num [1:2]	1 2
- Console (Bottom Left):** Shows the execution of the following commands:


```

> x <- 1
> y <- c(1,2)
> z <- matrix(1:24, nrow=4)
> w <- array(1:24, dim=c(2,3,4))
>

```
- Help Panel (Bottom Right):** Displays the documentation for the `array` function, titled "Multi-way Arrays". It includes a description, usage, and arguments.

Multi-way Arrays

Description
Creates or tests for arrays.

Usage
`array(data = NA, dim = length(data), dimnames = NULL)`
`as.array(x, ...)`
`is.array(x)`

Arguments

 - data**: a vector (including a list or [expression](#) vector) giving data to fill the array. Non-atomic classed objects are coerced by [as.vector](#).
 - dim**: the dim attribute for the array to be created, that is an integer vector of length one or more giving the maximal indices in each dimension.
 - dimnames**: either `NULL` or the names for the dimensions. This must be a list (or it will be ignored) with one component for each dimension, either `NULL` or a

7. DataFrame, List

데이터 종류: 데이터프레임 (DataFrame)

- 데이터프레임(DataFrame) : 동일하거나 다른 형태의 자료를 2차원 이상의 형태로 여러 개 모아서 취급하는 데이터 형태
 - 숫자와 문자가 섞여서 구성될 수 있음.
 - 예) 티커, 날짜, 주가의 데이터
- 데이터프레임 생성 방법
 - 배열 데이터를 (벡터 형태로) 선언한다.
 - 선언된 배열 데이터를 모아서 구성한다.

데이터 종류: 데이터프레임 (DataFrame)

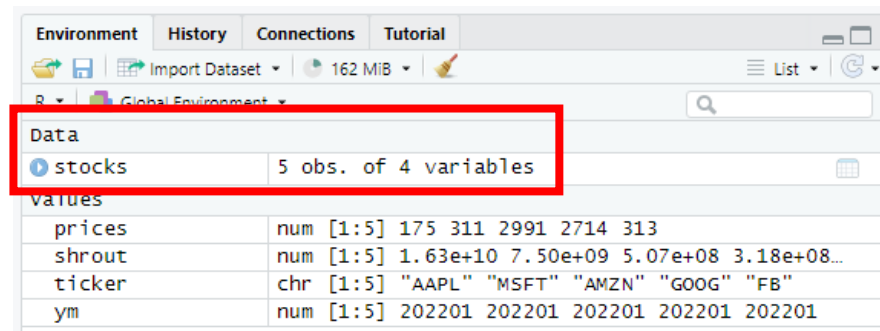
- 데이터프레임 (DataFrame) 생성

```
230 # Dataframe
231
232 ticker <- c("AAPL", "MSFT", "AMZN", "GOOG", "FB")
233 ym <- c(202201, 202201, 202201, 202201, 202201)
234 prices <- c(174.78, 310.98, 2991.47, 2713.97, 313.26)
235 shrout <- c(16334371000, 7496866000, 507148000, 317738000, 2366278000)
236
237 stocks <- data.frame(ticker = ticker, ym = ym, prices = prices, shrout = shrout)
238 stocks
239
240
```

240:1 (Top Level) ↕

Console Terminal x Jobs x

```
R 4.1.2 · ~/
> ticker <- c("AAPL", "MSFT", "AMZN", "GOOG", "FB")
> ym <- c(202201, 202201, 202201, 202201, 202201)
> prices <- c(174.78, 310.98, 2991.47, 2713.97, 313.26)
> shrout <- c(16334371000, 7496866000, 507148000, 317738000, 2366278000)
> stocks <- data.frame(ticker = ticker, ym = ym, prices = prices, shrout = shrout)
> stocks
  ticker    ym  prices    shrout
1  AAPL 202201  174.78 16334371000
2  MSFT 202201  310.98  7496866000
3  AMZN 202201 2991.47  507148000
4  GOOG 202201 2713.97  317738000
5   FB  202201  313.26  2366278000
>
```



Environment History Connections Tutorial

Import Dataset 162 MiB

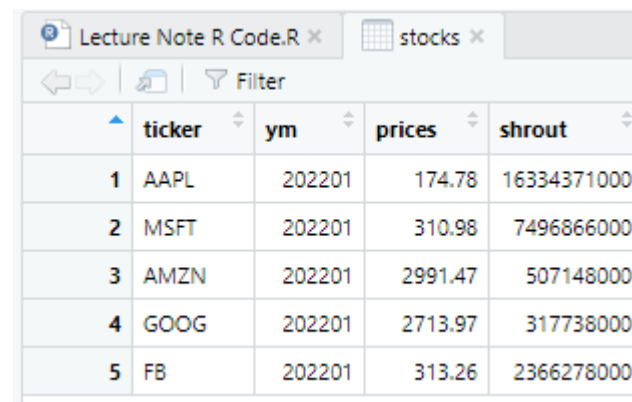
Global Environment

Data

stocks 5 obs. of 4 variables

values

prices	num [1:5]	175 311 2991 2714 313
shrout	num [1:5]	1.63e+10 7.50e+09 5.07e+08 3.18e+08...
ticker	chr [1:5]	"AAPL" "MSFT" "AMZN" "GOOG" "FB"
ym	num [1:5]	202201 202201 202201 202201 202201



Lecture Note R Code.R x stocks x

Filter

	ticker	ym	prices	shrout
1	AAPL	202201	174.78	16334371000
2	MSFT	202201	310.98	7496866000
3	AMZN	202201	2991.47	507148000
4	GOOG	202201	2713.97	317738000
5	FB	202201	313.26	2366278000

R studio!!!

데이터 종류: 데이터프레임 (DataFrame)

- 데이터프레임 (DataFrame) 특정 행과 열 출력
 - 이것을 하기위해서 matrix에 대한 indexing을 배웠습니다.

- Matrix처럼 숫자로 하는 indexing

- stocks[1]

```
> stocks[1]
  ticker
1  AAPL
2  MSFT
3  AMZN
4  GOOG
5   FB
```

- Dataframe에서는 column을 반환한다.

- stocks[1,]

```
> stocks[1,]
  ticker      ym prices      shrout
1  AAPL 202201 174.78 16334371000
```

- stocks[,1]

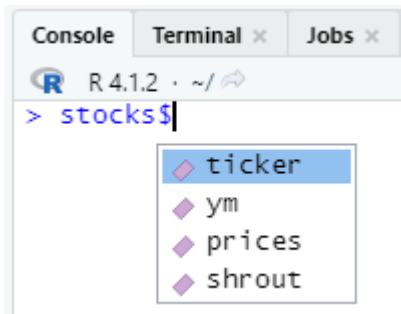
```
> stocks[,1]
[1] "AAPL" "MSFT" "AMZN" "GOOG" "FB"
```

- stocks[,c(1,3)]

```
> stocks[,c(1,3)]
  ticker prices
1  AAPL  174.78
2  MSFT  310.98
3  AMZN 2991.47
4  GOOG 2713.97
5   FB  313.26
```

데이터 종류: 데이터프레임 (DataFrame)

- Dataframe의 column name으로 하는 indexing
 - 이것을 하기위해서 dataframe을 사용합니다.



- Rstudio의 강점
 - 위와 같이 갖고 있는 column name을 알려준다.
- stocks\$ticker

```
> stocks$ticker
[1] "AAPL" "MSFT" "AMZN" "GOOG" "FB"
```
- stocks\$prices

```
> stocks$prices
[1] 174.78 310.98 2991.47 2713.97 313.26
```

데이터 종류: 데이터프레임 (DataFrame)

- Conditional indexing in Dataframe
 - 문제
 - 한 기업당 백만원 내에서 투자하려고 한다. (환율: 편의상 1,000원)
 - Universe에서 투자조건에 맞는 주식만 선별하시오.
 - 힌트
 - stocks에 []안에 조건문을 쓴다.
 - 조건
 - stocks' price is less than 1000 USD

데이터 종류: 데이터프레임 (DataFrame)

- Column간의 연산
 - 실제 가장 많이 활용하는 연산
 - 변수만들 때 가장 많이 활용
 - `stocks$prices * stocks$shrout`
 - 의미?
 - `stocks$mktcap <- stocks$price * stocks$shrout`
 - 생성한 변수는 꼭 이런식으로 dataframe에 추가시켜준다.
- [실습 17] 이번에는 universe에서 시가총액이 천조원 이상인 기업만 선별하려고 한다. 코드를 작성해 보시오.

Rstudio 유용한 팁

- ctrl + Enter를 사용하면 debugging하실때 매우 편리
 - 한줄에서 ctrl+Enter 을 눌러서 어떤 일이 생기는지 확인
- 블록을 설정하고 ctrl+Enter가능
 - 커서로 블록을 설정하고 ctrl+Enter 을 눌러서 어떤 일이 생기는지 확인

r install

검색어

R스튜디오

주제

+ 비교 추가

미국

2004 - 현재

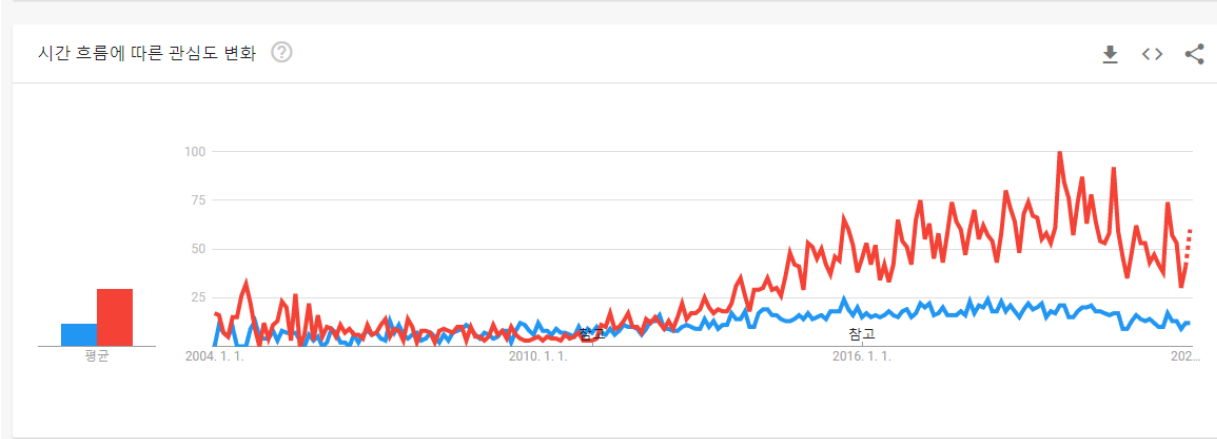
모든 카테고리

웹 검색

!

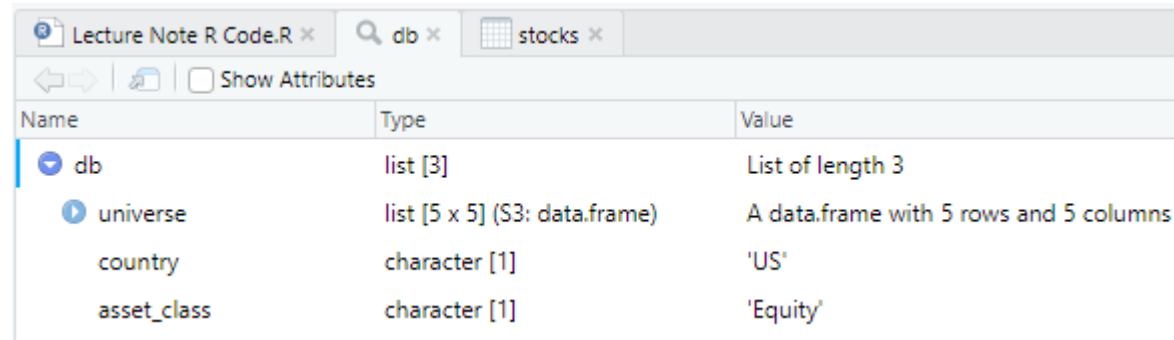
주의: 이 비교에는 검색어와 주제가 모두 포함되어 있으며, 이 두 항목은 서로 다르게 측정됩니다.

자세히 알아보기



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.**
 - 사실 앞의 dataframe을 생성하는 과정은 실전 데이터 분석에서는 거의 할 일이 거의 없음
 - 대부분 **DB화 되어 있는 데이터**를 불러옴
 - 불러온 데이터는 **list 형태로** 주는 것이 업계표준
 - 자유도가 아주 높음
 - 대신 협업을 위해서 **가독성이 높게 데이터를 전달**해야함
 - `db <- list(universe=stocks, country="US", asset_class="Equity")`
 - 각 key에 item(데이터)를 저장
 - python dictionary랑 유사



The screenshot shows the R Studio interface with a list object named 'db' in the Environment pane. The list contains three elements: 'universe' (a data frame with 5 rows and 5 columns), 'country' (a character vector with value 'US'), and 'asset_class' (a character vector with value 'Equity').

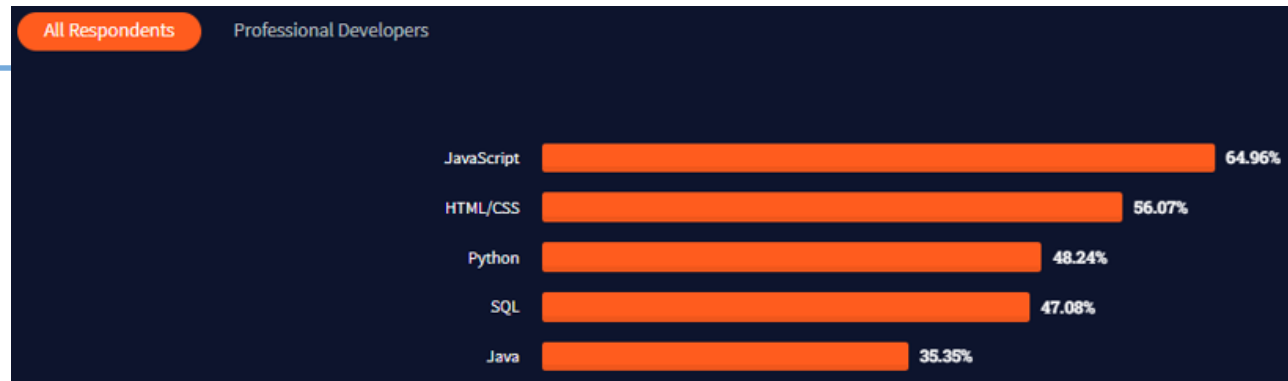
Name	Type	Value
db	list [3]	List of length 3
universe	list [5 x 5] (S3: data.frame)	A data.frame with 5 rows and 5 columns
country	character [1]	'US'
asset_class	character [1]	'Equity'

데이터 종류: 리스트 (List)

- List에 access하기
 - db\$universe
 - db\$country
 - db\$asset_class
- List에 key 추가
 - db\$author <- “Cho”
- List에 key 제거
 - db\$author <- NULL

Dataframe으로 데이터 작업하는 습관들이기

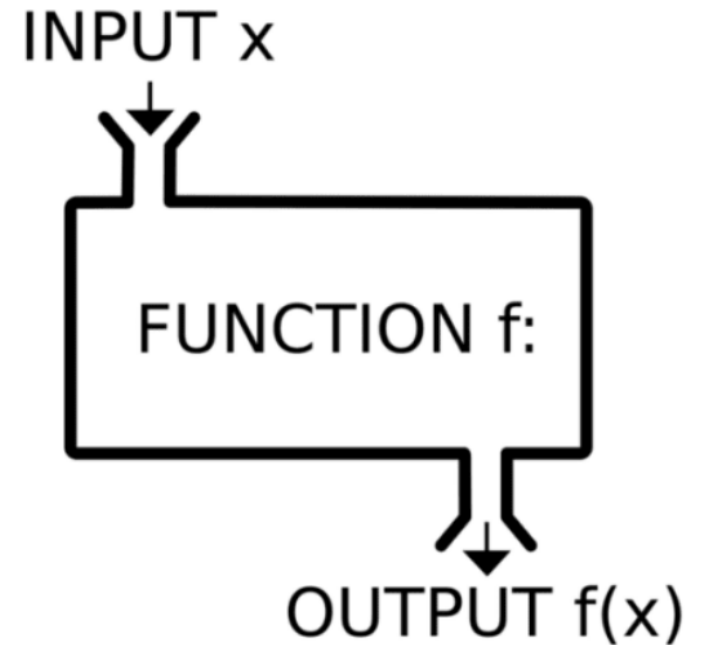
- Matrix는 실수하기 쉬움
 - Indexing이 주로 숫자로 이뤄짐
- Dataframe은 그 실수를 줄여줄 수 있음
 - Column name으로 주로 access하기 때문에 논리흐름에 따라 코딩가능
 - SQL 스타일의 코딩이 가능
 - SQL: 질의 -> 응답
 - “어떤 조건을 만족하는 데이터를 줘”하면 그에 해당하는 데이터 반환
 - R의 dataframe으로 하는 conditional indexing이 결국 SQL임
 - 이러한 쿼리형 데이터는 작업할 때 논리의 흐름을 파악하기 더 쉬워서 여러모로 편리
- 나아가 R의 list, python의 dictionary 형태에 익숙해시면 좋습니다.
 - 요즘 어플이나 각종 소비자행태 데이터는 모두 JSON (JavaScript Object Notation)으로 쌓임
 - JSON이 다 R의 list나 python dictionary처럼 key-item의 매칭인 relational database임.



8. 함수, 반복문, 그리고 조건문 데이터 작업을 현명하게 수행하기

함수 (Function)

- 데이터 작업할 때 실수를 줄여주고 효율을 높여줌.
 - 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)

- 예제.

- 유니버스마다 시가총액을 계산하고 그 중 1000조원보다 큰 스탁만 가져오려고 한다.
- 이때 10개의 유니버스가 주어졌다면 함수를 안 쓰면 다음과 같이 해야함.

```
universe1$mktcap <- universe1$prices*universe1$shrout
universe1_final <- universe1[universe1$mktcap>1e12,]
universe2$mktcap <- universe2$prices*universe2$shrout
universe2_final <- universe2[universe2$mktcap>1e12,]
universe3$mktcap <- universe3$prices*universe3$shrout
universe3_final <- universe3[universe3$mktcap>1e12,]
universe4$mktcap <- universe4$prices*universe1$shrout
universe4_final <- universe4[universe1$mktcap>1e12,]
universe5$mktcap <- universe5$prices*universe1$shrout
universe5_final <- universe5[universe1$mktcap>1e12,]
universe6$mktcap <- universe6$prices*universe1$shrout
universe6_final <- universe6[universe1$mktcap>1e12,]
universe7$mktcap <- universe7$prices*universe7$shrout
universe7_final <- universe7[universe1$mktcap>1e12,]
universe8$mktcap <- universe8$prices*universe8$shrout
universe8_final <- universe8[universe1$mktcap>1e12,]
universe9$mktcap <- universe9$prices*universe9$shrout
universe9_final <- universe9[universe9$mktcap>1e12,]
universe10$mktcap <- universe10$prices*universe10$shrout
universe10_final <- universe10[universe10$mktcap>1e12,]
```

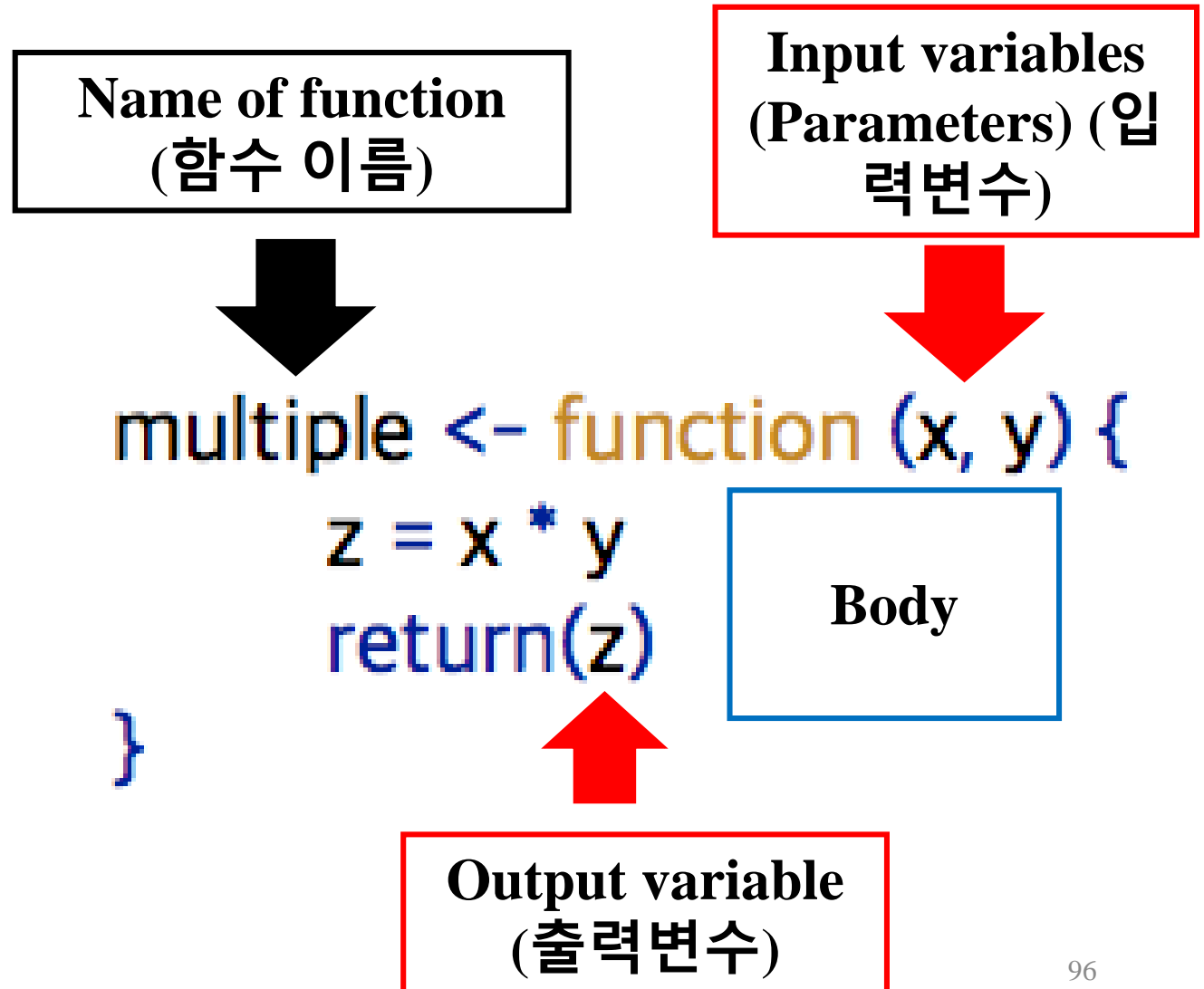
- 함수를 써야함

함수 (Function)

- 함수를 다음과 같이 작성

```
get_universe <- function(df){  
  df$mktpcap <- df$prices*df$shrout  
  df_final <- df[df$mktpcap>1e12,]  
  return(df_final)  
}
```

Functions	
get_universe	function (df)



함수 (Function)

- 함수를 써서 아까의 코딩을 한다면?

```
universe1$mktcap <- universe1$prices*universe1$shrout
universe1_final <- universe1[universe1$mktcap>1e12,]
universe2$mktcap <- universe2$prices*universe2$shrout
universe2_final <- universe2[universe2$mktcap>1e12,]
universe3$mktcap <- universe3$prices*universe3$shrout
universe3_final <- universe3[universe3$mktcap>1e12,]
universe4$mktcap <- universe4$prices*universe4$shrout
universe4_final <- universe4[universe4$mktcap>1e12,]
universe5$mktcap <- universe5$prices*universe5$shrout
universe5_final <- universe5[universe5$mktcap>1e12,]
universe6$mktcap <- universe6$prices*universe6$shrout
universe6_final <- universe6[universe6$mktcap>1e12,]
universe7$mktcap <- universe7$prices*universe7$shrout
universe7_final <- universe7[universe7$mktcap>1e12,]
universe8$mktcap <- universe8$prices*universe8$shrout
universe8_final <- universe8[universe8$mktcap>1e12,]
universe9$mktcap <- universe9$prices*universe9$shrout
universe9_final <- universe9[universe9$mktcap>1e12,]
universe10$mktcap <- universe10$prices*universe10$shrout
universe10_final <- universe10[universe10$mktcap>1e12,]
```

- 절반으로 줄어듦

- 일반적으로 1/n로 줄일 수 있음

```
universe1_final1 <- get_universe(universe1)
universe1_final2 <- get_universe(universe2)
universe1_final3 <- get_universe(universe3)
universe1_final4 <- get_universe(universe4)
universe1_final5 <- get_universe(universe5)
universe1_final6 <- get_universe(universe6)
universe1_final7 <- get_universe(universe7)
universe1_final8 <- get_universe(universe8)
universe1_final9 <- get_universe(universe9)
universe1_final10 <- get_universe(universe10)
```

- 반복문을 활용해서 좀더 체계적으로 접근할 수 있음.

반복문 (for 문)

- 반복문 예시

```
> for (i in 1:10){  
+   print(i)  
+ }  
[1] 1  
[1] 2  
[1] 3  
[1] 4  
[1] 5  
[1] 6  
[1] 7  
[1] 8  
[1] 9  
[1] 10
```

- 반복문 문법

- for (i in 인덱스범위){

반복 논리

}

```
universes <- list(stocks, stocks, stocks, stocks, stocks,  
                 stocks, stocks, stocks, stocks, stocks  
                 )  
list_universe = list()  
for (i in 1:10){  
  tmp_df <- universes[[i]]  
  tmp_universe <- get_universe(tmp_df)  
  tmp_list <- list(tmp_universe)  
  list_universe <- append(list_universe, tmp_list)  
}  
list_universe
```

조건문

- 만약 6번째 유니버스부터는 전처리를 하고싶지 않다면?
- 조건문 문법
 - if (조건){
 작업
}

```
universes <- list(stocks, stocks, stocks, stocks, stocks,  
                 stocks, stocks, stocks, stocks, stocks  
)  
list_universe = list()  
for (i in 1:10){  
  if (i<=5){  
    tmp_df <- universes[[i]]  
    tmp_universe <- get_universe(tmp_df)  
    tmp_list <- list(tmp_universe)  
    list_universe <- append(list_universe, tmp_list)  
  }  
}  
list_universe
```

조건문

- [실습 19] 만약 6번째 유니버스부터는 전처리를 하고싶지 않되 10번째는 하고 싶다면?

조건문

- else if, else 구문
 - 예) 10번째는 알고봤더니 지금 전처리를 하면 안되지만 나중에 살펴볼만하긴함

```
universes <- list(stocks, stocks, stocks, stocks, stocks,
                  stocks, stocks, stocks, stocks, stocks)
)
list_universe = list()
for (i in 1:10){
  if (i<=5){
    tmp_df <- universes[[i]]
    tmp_universe <- get_universe(tmp_df)
    tmp_list <- list(tmp_universe)
    list_universe <- append(list_universe, tmp_list)
  } else if (i==10){
    print(i)
    print('Discuss later')
  } else {
    print(i)
    print('No.')
  }
}

list_universe
length(list_universe)
```

```
[1] 6
[1] "No."
[1] 7
[1] "No."
[1] 8
[1] "No."
[1] 9
[1] "No."
[1] 10
[1] "Discuss later"
```

- 함수, 반복문, 조건문을 잘 활용하여 현명한 데이터 작업 및 통계분석을 해야한다.

-
- ifelse 함수
 - 데이터 전처리에 유용
 - `x <- runif(1); x` # 0과 1 사이의 난수를 생성해서 x에 저장
 - `ifelse(x>0.5, "Large", "Small")` #x가 0.5보다 크면 Large 출력, 0.5보다 작으면 Small 출력
 - `x <- runif(10); x`
 - `ifelse(x>0.5, "Large", "Small")`
 - vectorize 연산
 - 가능한 경우가 많으니 꼭 체크하고 효율적인 방식으로 작업

9. 본격적인 데이터 작업 Part 1

패키지 설치, 데이터 불러와서 다루기

패키지 (Package)

- 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

Install Package

- 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.

```
# Package install and load

##Required packages
#데이터 읽어오기
install.packages('data.table')
install.packages('readxl')
#데이터 전처리
install.packages('plyr')
install.packages('dplyr')
#데이터 합치기
install.packages('bindrcpp')
#데이터 형태 바꾸기
install.packages('reshape')
#데이터 그래프 그리기
install.packages('ggplot2')
install.packages('Rcpp')
```

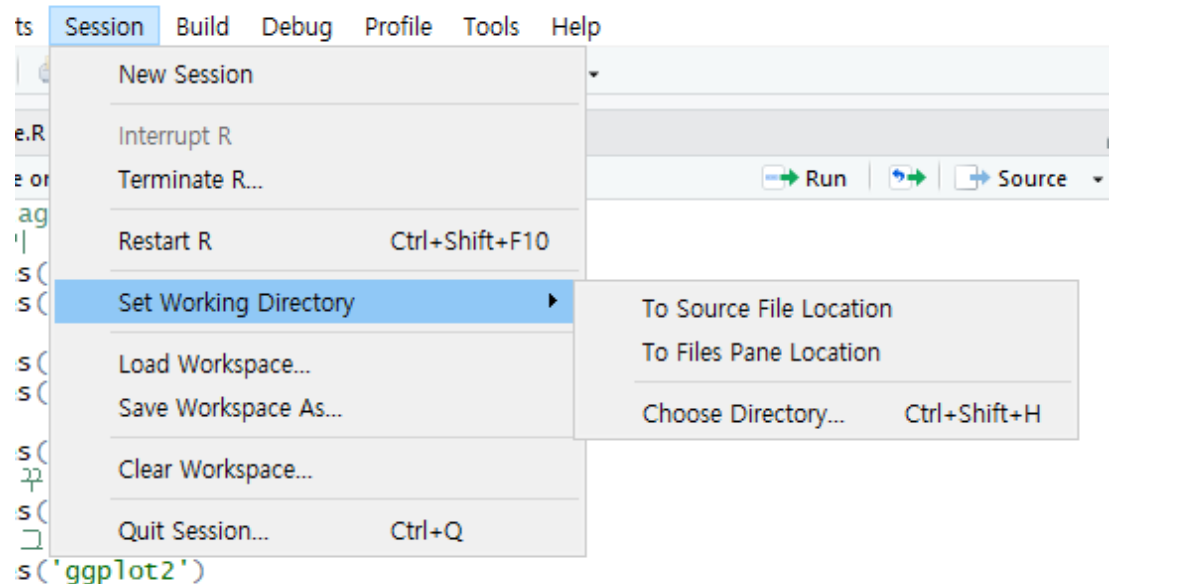
Load Library

- 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.

```
#라이브러리 로드  
library('data.table')  
library('readxl')  
library('plyr')  
library('dplyr')  
library('bindrcpp')  
library('readr')
```

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.

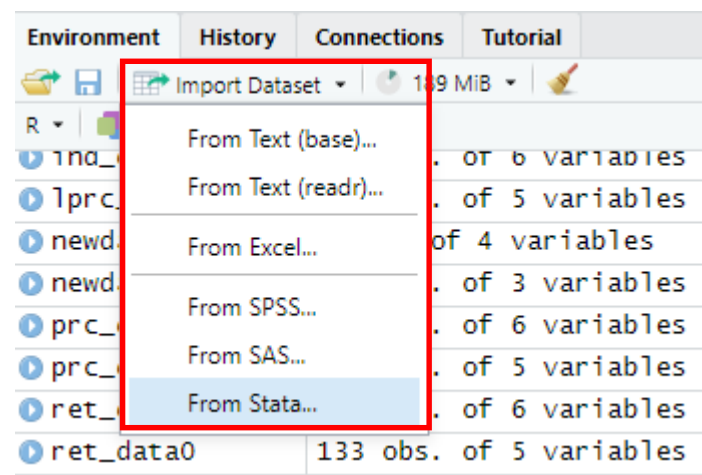


```
#working directory 설정  
getwd()  
setwd('C:/Users/FeeLab/Dropbox/TA/DFMBA 2022/data')
```

데이터 로드

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

```
135 #data load
136 ind_data <- read_excel('kospi dataset.xlsx',
137                        sheet = 'mktcap&ind code')
138
139 rm(ind_data)
140
141 ind_data <- read_csv('kospi mktcap&ind code.csv',
142                    head = TRUE)
143
```



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
>
> 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
```

```
> 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
```

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) 방
향으로 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
```

10. 본격적인 데이터 작업 Part 2

탐색적 데이터 분석 – 도식화

변수의 종류

변수 분류	변수 종류	설명
Categorical variables (Qualitative)	Nominal data	숫자로 표시된 데이터가 아니며, 편의상 숫자로 변환. 순위의 개념이 없음 (예: 산업 코드)
	Ordinal data	숫자로 표시된 데이터가 아니며, 편의상 숫자로 변환. 순위의 개념이 있음 (예: 시가총액의 분류)
Numeric variables (Quantitative)	Continuous data	데이터가 연속 변수로써 셀 수 없는 형태 (예: 시가총액, 주가)
	Discrete data	데이터가 비연속 변수로써 셀 수 있는 형태 (예: 기업의 직원 수)

변수의 종류에 따른 표현 방법

변수 분류	표현의 종류	종류
Categorical variables (Qualitative)	표 형태의 표현	빈도표, 상대빈도표, 퍼센트 빈도표, 교차 테이블
	그림 형태의 표현	바 차트, 파이 차트, 그룹 바 차트, 누적 차트
Numeric variables (Quantitative)	표 형태의 표현	빈도표, 상대적 빈도표, 퍼센트 빈도표, 누적 빈도표, 교차 테이블
	그림 형태의 표현	히스토그램, 줄기-잎 도표, 스캐터 플랏

데이터 분석: Categorical variables

- 카테고리 변수는 원칙적으로 숫자로 표시할 수 없으나, 편의상 숫자로 변경한 데이터를 의미함.
 - 기업별 산업 코드는 대표적인 카테고리 변수로 숫자의 순서가 의미가 없는 변수임.
 - `ind_data$ind_code <- as.numeric(factor(ind_data$industry))`

	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
13	A028260	삼성물산	27285514	제조업
14	A096770	SK이노베이션	26491384	제조업
15	A051900	LG생활건강	26176098	제조업
16	A034730	SK	24977905	제조업
17	A005490	POSCO	23060918	제조업
18	A036570	엔씨소프트	21844252	제조업
19	A017670	SK텔레콤	21236122	제조업
20	A003550	LG	19067563	제조업



	symbol	name	mktcap	industry	ind_code
1	A005930	삼성전자	533698560	제조업	5
2	A000660	SK하이닉스	98280319	제조업	5
3	A051910	LG화학	69886420	제조업	5
4	A035420	NAVER	57327925	제조업	5
5	A005380	현대차	55553729	제조업	5
6	A006400	삼성SDI	54323979	제조업	5
7	A207940	삼성바이오로직스	52270350	제조업	5
8	A068270	셀트리온	43334295	제조업	5
9	A035720	카카오	41905696	제조업	5
10	A000270	기아차	37820400	제조업	5
11	A012330	현대모비스	33031506	제조업	5
12	A066570	LG전자	29129311	제조업	5
13	A028260	삼성물산	27285514	제조업	5
14	A096770	SK이노베이션	26491384	제조업	5
15	A051900	LG생활건강	26176098	제조업	5
16	A034730	SK	24977905	제조업	5
17	A005490	POSCO	23060918	제조업	5
18	A036570	엔씨소프트	21844252	제조업	5
19	A017670	SK텔레콤	21236122	제조업	5
20	A003550	LG	19067563	제조업	5

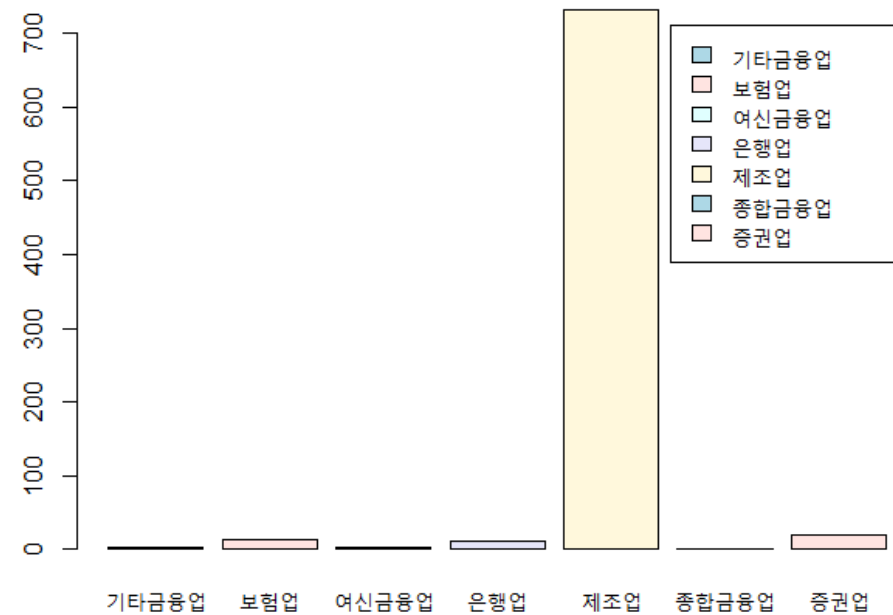
데이터 분석: Categorical variables

- 카테고리 변수 빈도 확인
 - `freq_table01 <- table(ind_data$industry)` # category 별 분할표를 만들
 - `freq_table1 <- as.data.frame(freq_table01)` # 다른 column도 추후 지정해주기 위해 dataframe으로 변환
 - `colnames(freq_table1) <- c('cat','freq')`
 - `freq_table1$ratio <- freq_table1$freq / sum(freq_table1$freq)`

	cat	freq	ratio
1	기타금융업	2	0.002567394
2	보험업	12	0.015404365
3	여신금융업	2	0.002567394
4	은행업	11	0.014120668
5	제조업	732	0.939666239
6	종합금융업	1	0.001283697
7	증권업	19	0.024390244

데이터 분석: Categorical variables

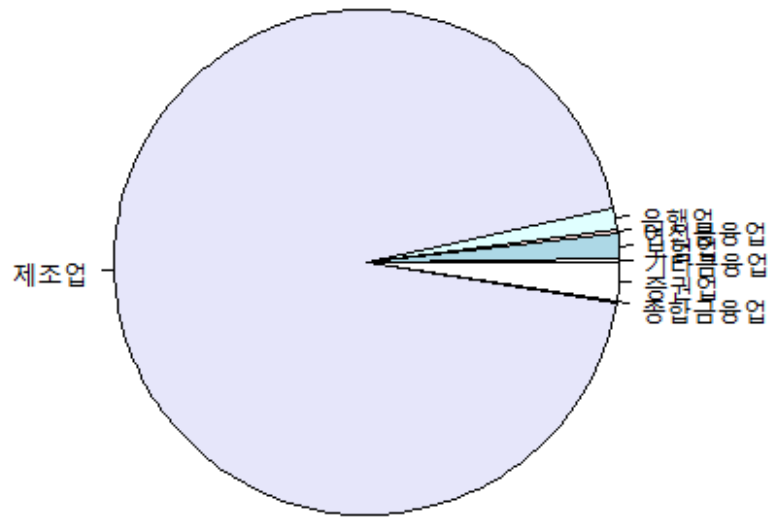
- 카테고리 변수 막대 차트 그리기
 - `col_list <- c("lightblue", "mistyrose", "lightcyan", "lavender", "cornsilk")`
 - `barplot(freq_table01, col = col_list, legend = TRUE)`



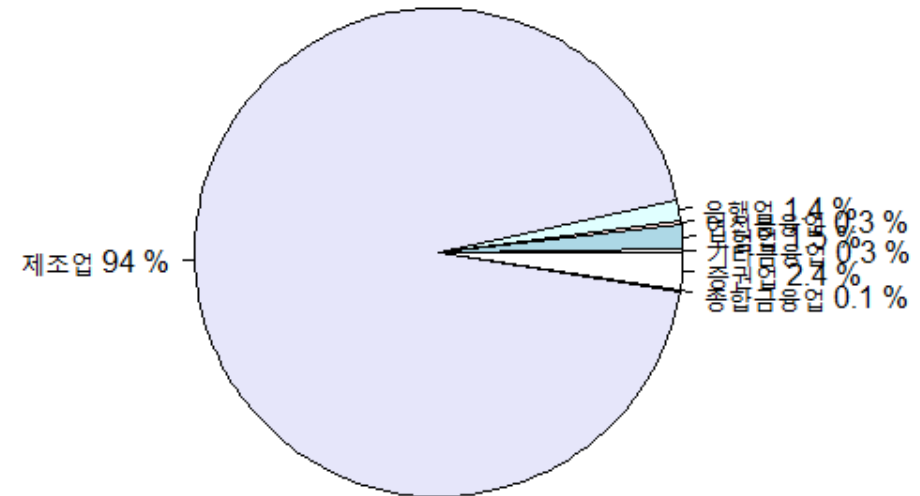
데이터 분석: Categorical variables

- 카테고리 변수 파이 차트 그리기

`pie(freq_table0)`



```
pie(freq_table0, labels = paste(freq_table$Var,  
                                round(freq_table$ratio * 100, 1), '%'))
```



데이터 분석: Categorical variables

- 시가총액을 기준으로 카테고리 변수로 변환
 - `mc <- ind_data$mktcap`
 - `mktcap_cart <- ifelse(mc >= 10000000, 'Large',
ifelse(mc < 10000000 & mc >= 1000000, 'Large-Medium',
ifelse(mc < 1000000 & mc >= 500000, 'Medium',
ifelse(mc < 500000 & mc >= 100000, 'Medium-Small',
'Small'))))`
 - `ind_data$mktcap_cart <- mktcap_cart`

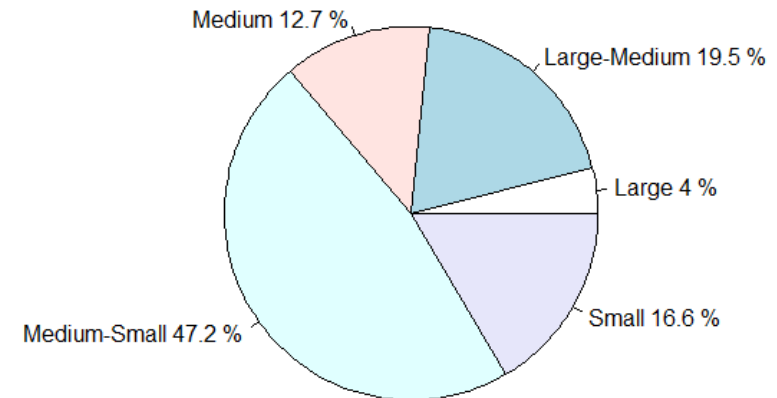
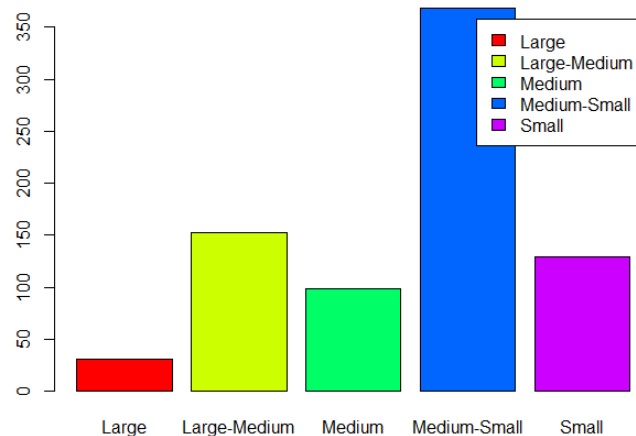
	symbol	name	mktcap	industry	ind_code	mktcap_cart
1	A005930	삼성전자	533698560	제조업	5	Large
2	A000660	SK하이닉스	98280319	제조업	5	Large
3	A051910	LG화학	69886420	제조업	5	Large
4	A035420	NAVER	57327925	제조업	5	Large
5	A005380	현대차	55553729	제조업	5	Large
6	A006400	삼성SDI	54323979	제조업	5	Large
7	A207940	삼성바이오로직스	52270350	제조업	5	Large
8	A068270	셀트리온	43334295	제조업	5	Large
9	A035720	카카오	41905696	제조업	5	Large
10	A000270	기아차	37820400	제조업	5	Large
11	A012330	현대모비스	33031506	제조업	5	Large
12	A066570	LG전자	29129311	제조업	5	Large
13	A028260	삼성물산	27285514	제조업	5	Large
14	A096770	SK이노베이션	26491384	제조업	5	Large
15	A051900	LG생활건강	26176098	제조업	5	Large
16	A034730	SK	24977905	제조업	5	Large
17	A005490	POSCO	23060918	제조업	5	Large
18	A036570	엔씨소프트	21844252	제조업	5	Large
19	A017670	SK텔레콤	21236122	제조업	5	Large
20	A003550	LG	19067563	제조업	5	Large

데이터 분석: Categorical variables

- [실습 20]
 - 다음과 같이 Large, Large-Medium 등의 시가총액 별 카테코리에 대한 요약통계치 표를 만드시오.

	cart	freq	ratio
1	Large	31	0.03979461
2	Large-Medium	152	0.19512195
3	Medium	99	0.12708601
4	Medium-Small	368	0.47240051
5	Small	129	0.16559692

- 다음과 같이 histogram과 pie차트를 그려보시오.



데이터 분석: Categorical & Categorical

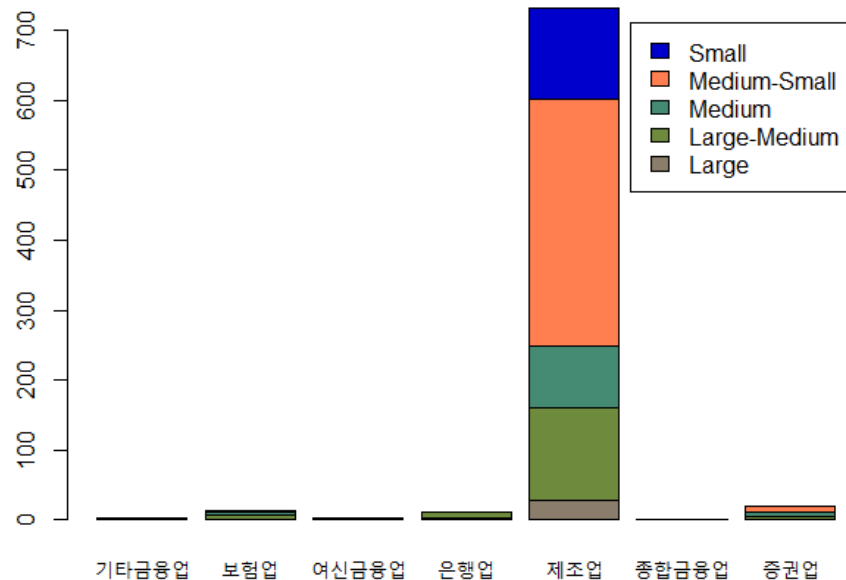
- 두 개의 Categorical 변수에 대한 **교차 테이블** 작성
 - `freq_table03 <- table(ind_data$industry, ind_data$mktcap_cart)` # 두개 인수를 통해 교차테이블
 - `freq_table3 <- as.data.frame(freq_table03)`
 - `colnames(freq_table3) <- c('ind','mktcap','freq')`
 - `freq_table4 <- unstack(freq_table3, freq~ind)`
 - `rownames(freq_table4) <- freq_table2$cart`
 - `freq_table5 = freq_table4/sum(freq_table4) * 100`

	기타 금융 업	보험 업	여신 금융 업	은행 업	제조 업	종합 금융 업	증권 업
Large	0	1	0	3	27	0	0
Large-Medium	0	5	1	7	134	0	5
Medium	1	4	1	0	88	0	5
Medium-Small	1	2	0	1	354	1	9
Small	0	0	0	0	129	0	0

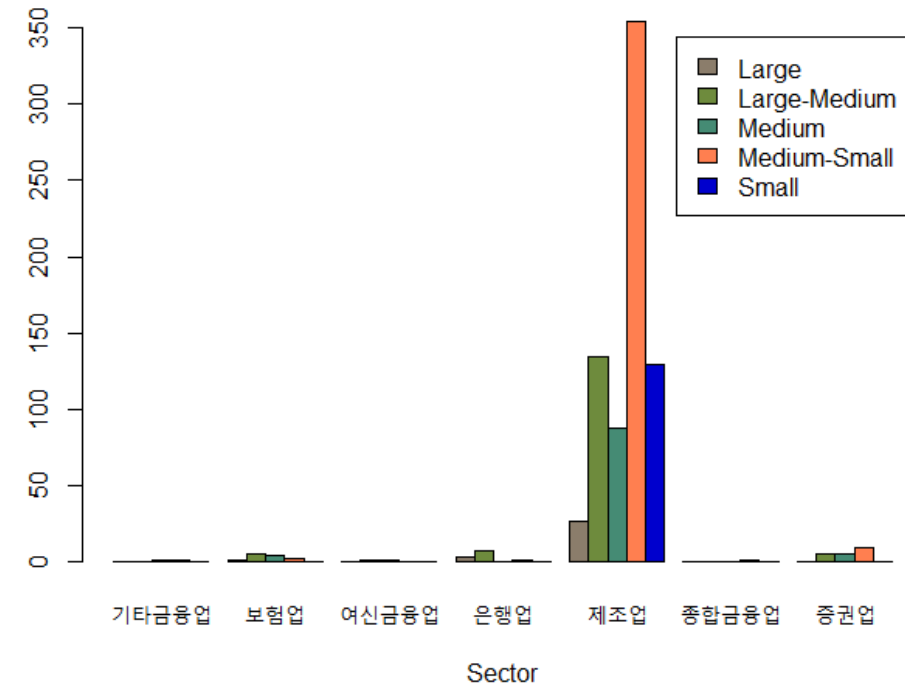
	기타 금융 업	보험 업	여신 금융 업	은행 업	제조 업	종합 금융 업	증권 업
Large	0.0000000	0.1283697	0.0000000	0.3851091	3.465982	0.0000000	0.0000000
Large-Medium	0.0000000	0.6418485	0.1283697	0.8985879	17.201540	0.0000000	0.6418485
Medium	0.1283697	0.5134788	0.1283697	0.0000000	11.296534	0.0000000	0.6418485
Medium-Small	0.1283697	0.2567394	0.0000000	0.1283697	45.442875	0.1283697	1.1553273
Small	0.0000000	0.0000000	0.0000000	0.0000000	16.559692	0.0000000	0.0000000

데이터 분석: Categorical & Categorical

- `barplot(data.matrix(freq_table4),`
- `col = colors()[c(23,89,12,57,29)],`
- `legend = rownames(freq_table5))`



- `barplot(data.matrix(freq_table4),`
- `col = colors()[c(23,89,12,57,29)], beside = T,`
- `xlab = 'Sector', legend = rownames(freq_table5))`



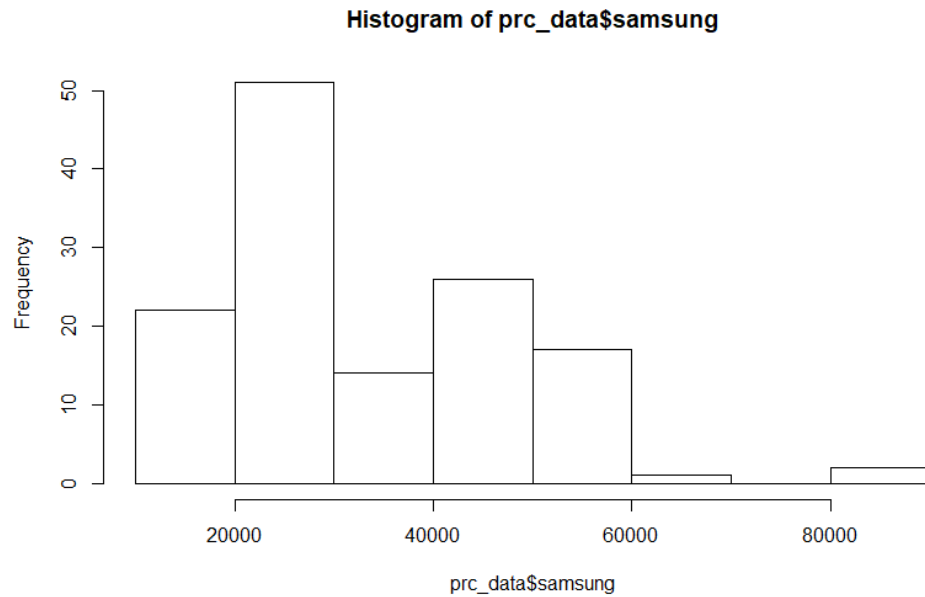
데이터 분석: Continuous variables

- 연속 변수는 연속된 숫자로 된 변수
 - 예를 들어, 기업의 월별 주가
- `prc_data <- read.csv('koshi stock price.csv', head = TRUE)`

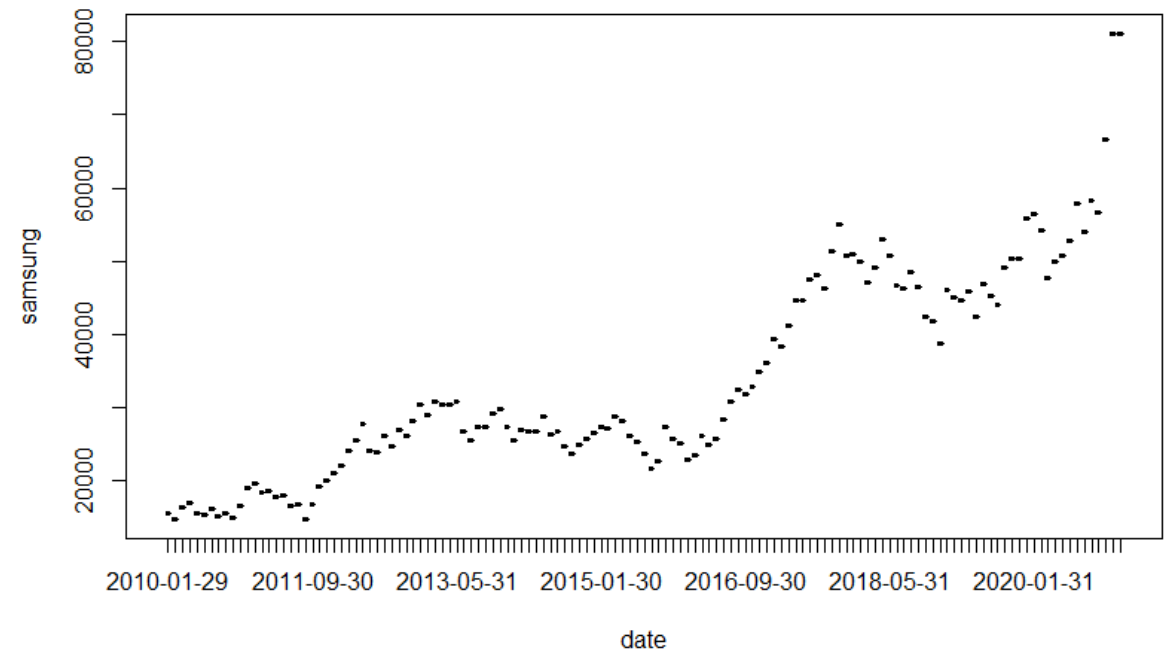
	date	samsung	sk.hynix	lg.chem	hyundai.motors	KOSPI
1	2010-01-29	15680	22750	200000	113000	1602.43
2	2010-02-26	14880	21000	215000	115000	1594.58
3	2010-03-31	16360	26700	240500	115500	1692.85
4	2010-04-30	16980	28400	283000	137000	1741.56
5	2010-05-31	15520	25150	273000	140000	1641.25
6	2010-06-30	15480	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
11	2010-11-30	16520	23500	388000	172500	1904.63
12	2010-12-30	18980	24000	391000	173500	2051.00
13	2011-01-31	19620	29650	420000	179000	2069.73
14	2011-02-28	18460	28350	372000	178000	1939.30
15	2011-03-31	18640	31300	460000	203000	2106.70
16	2011-04-29	17860	33800	530000	246500	2192.36
17	2011-05-31	18040	30100	535000	252500	2142.47
18	2011-06-30	16520	25050	488000	237000	2100.69
19	2011-07-29	16880	24250	470000	235000	2133.21
20	2011-08-31	14880	19100	378500	203000	1880.11

데이터 분석: Continuous variables

- 연속 변수의 히스토그램
- `hist(prc_data$samsung)`



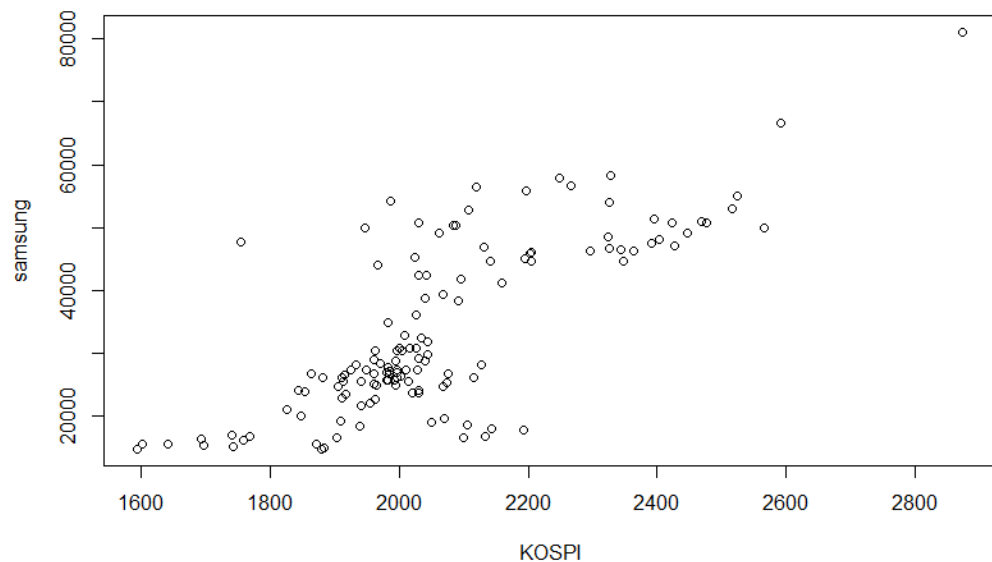
- `plot(samsung~date, prc_data)`



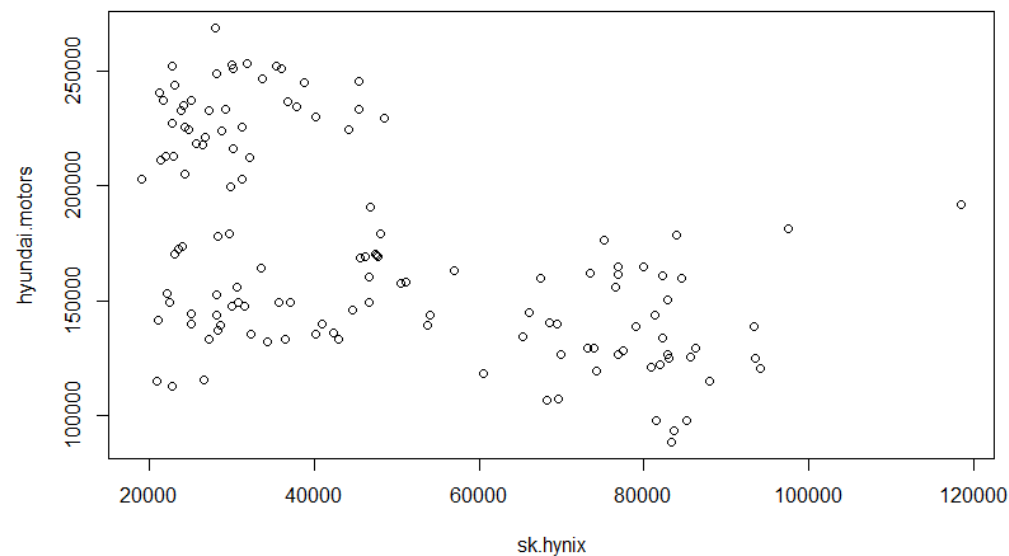
데이터 분석: Continuous variables

- 연속된 두 변수 간의 그림: 산점도 (Scatter plot)

- `plot(samsung~KOSPI, prc_data)`

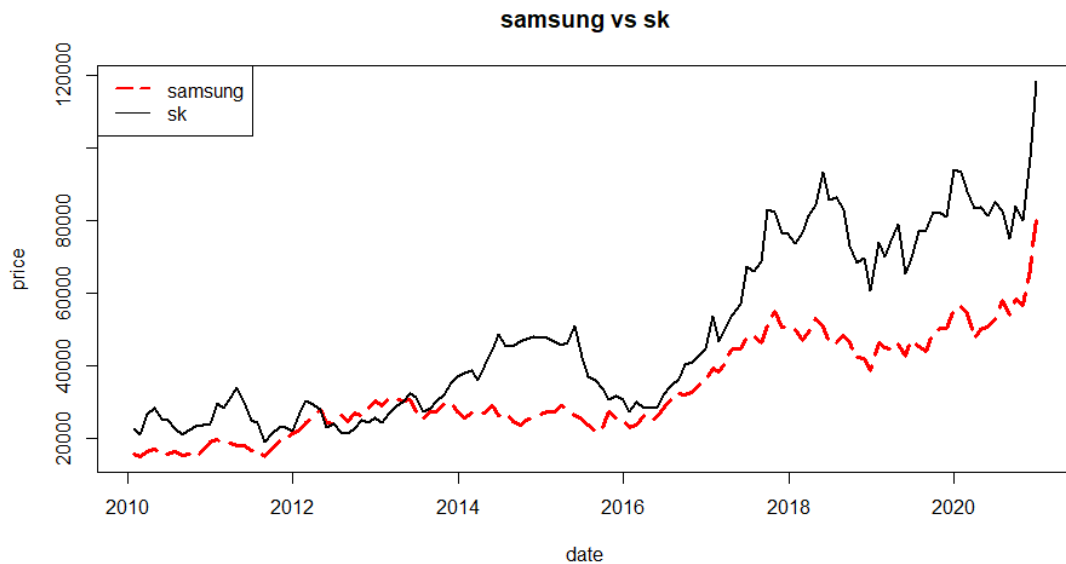


- SK하이닉스와 현대차 주가의 산점도



데이터 분석: Continuous variables

- 연속된 두 변수 간의 그림: multiple plot



- 다음의 그래프를 그려보시오.

- Hint: lty 3 & 4, lwd = 2, 'bottomright'
- lty: line type
- lwd: line width
- bottomright: legend position

