

R-입문

서울대학교 통계연구소

2023년 8월

이번 강의에서 다룰 내용

- R 이란?
- R 설치
- R Studio 소개 및 설치
- R package 설치
- R에서 데이터 종류 및 자료구조
- 간단한 데이터 조작

R이란?

R은 계산과 그래픽을 위한 프로그래밍 언어이자 소프트웨어 환경이다.

- 1990년대 초 뉴질랜드의 통계학자 Ross Ihaka와 Robert Gentleman에 의해 시작되었다.
- 기존에 Bell Labs에서 개발한 통계프로그램인 S와 유사하다.
- 무료로 사용 가능하고 누구나 패키지를 다운받아 사용하거나 직접 개발할 수도 있다.
- 통계학자들뿐만 아니라 여러 종류의 데이터를 다루는 기업들도 R을 많이 사용하게 되었다.
- 여러가지 통계적 기법을 수행하기 위한 함수가 많이 내장되어 있다.

- R은 윈도우, 맥, 리눅스등 다양한 운영체제에서 이용 가능하다.
- R은 커맨드라인 (Command line) 기반의 프로그램이다.
- 대부분의 작업이 잘 정의된 텍스트 명령어로 이루어진다.
- 정해진 문법에 따라 명령어가 작성되어야 한다.
- 결과가 잘못 되었을 때 제시되는 피드백이나 에러메세지가 아주 친절하지는 않다.
- 통계, 기계학습, 금융, 생물정보학, 그래픽스에 이르는 다양한 패키지가 무료로 제공된다.
- 방대한 온라인 매뉴얼이 있다. ([Stack Overflow](#), [Quick R](#))

R 배우기

- 컴퓨터, 프로그래밍, 데이터과학 초심자
 - ▶ 강력한 무료 데이터 분석도구 사용의 첫 삽
- 스프레드시트, 통계패키지, 회계소프트웨어 유경험자
 - ▶ 기존의 GUI, 마우스클릭과 작별하자.
- 사용자가 분석할 데이터에 대해 확실히 이해하고 있지 않다면 R 명령어가 제대로 동작하지 않을때 이유를 알기 어렵다.
- 데이터로 무엇을 할 수 있는지 없는지, 데이터를 어떻게 변환해야 할지, 분석시에 어떤 문제가 발생할 수 있을지 알아야 함. 즉, 사용자로 하여금 문제를 데이터 중심으로 보도록 유도함.
- R로 작업하면서 습득한 교훈은 다른 프로그램 또는 환경에 거의 언제나 적용 가능

R 설치하기: 윈도우 PC

① Comprehensive R Archive Network (CRAN)

- “Download and install R” 섹션에서 본인의 OS에 맞는 링크 클릭.



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

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 (2021-05-18, Camp Pontanezen) [R-4.1.0.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.

② “Subdirectories” 섹션에 “base” 링크 클릭.

R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to install R for the first time .
contrib	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R 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.

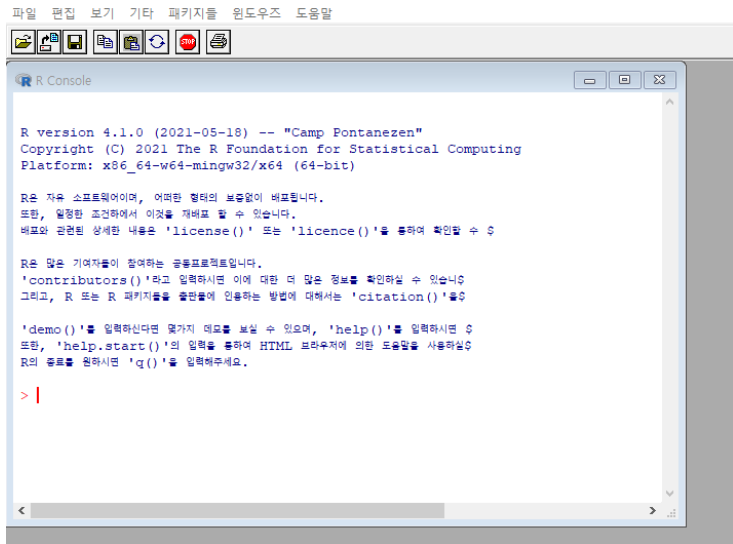
You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

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

③ 다음 페이지에서 “Download R X.X.X for Windows” 링크 클릭 (X.X.X 는 R 버전을 의미함. 예를 들어 R 4.2.2). 파일을 실행 또는 저장 후 실행. 설치에 수 분 소요.

R 실행하기

- “R” 또는 “R X.X.X”을 선택하여 “R” 실행. R 콘솔이 나타남

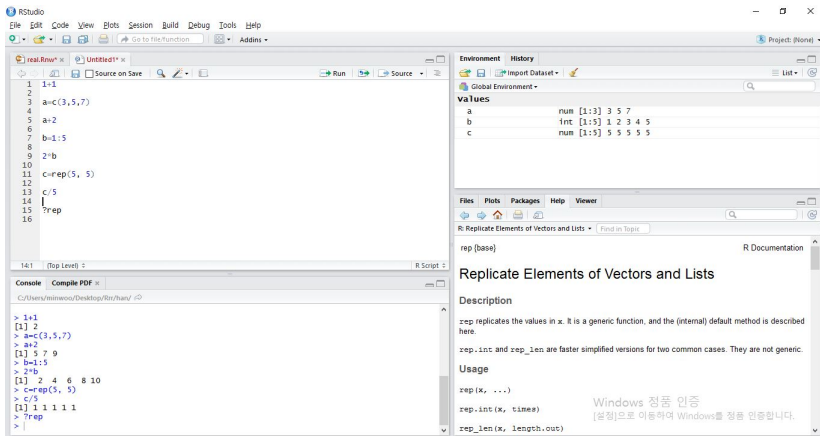


“콘솔”이란 컴퓨터가 집 한 채 크기던 시절 집 한켠에 모시고 거대한 기계를 조정하기 위한 “마스터 제어판”을 이르던 말. ([source link](#))



- R을 위한 통합환경 (IDE - Integrated Development Environment)
- 윈도우 응용 프로그램 개발에서의 Visual Studio와 비슷한 역할
- R 외에 Python 등 타 언어 코딩과 Markdown 등의 코딩을 수반하는 문서작업도 할 수 있는 종합 플랫폼으로 발전 중 (Posit)
- R 사용에 필요한 부가 정보들을 한 눈에 보여줌
- 불러오기, 저장하기 등 기본적인 설정은 직접 코딩하지않고 단순 클릭으로 실행할 수 있음

- R Studio에는 총 4개의 창이 나타나며 사용자 편의에 맞게 조절도 가능



R Studio 설치하기

- ① posit.co 에 접속
- ② 우측 상단의 “DOWNLOAD RSTUDIO” 클릭
- ③ 사용자 컴퓨터에 맞는 설치 파일을 선택하여 다운로드 후 설치

[PRODUCTS](#) ▾[SOLUTIONS](#) ▾[LEARN & SUPPORT](#) ▾[EXPLORE MORE](#) ▾[DOWNLOAD RSTUDIO](#)

Download the RStudio IDE

The most popular coding environment for R, built with love by Posit.

Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution. It also features tools for plotting, viewing history, debugging and managing your workspace.

[RStudio Desktop](#)[RStudio Server](#)

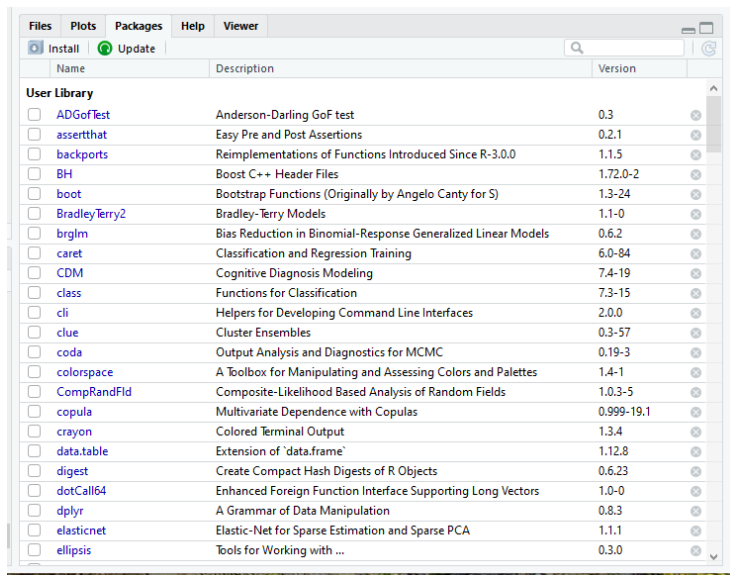
R and R Studio 튜토리얼

- 구글 또는 유튜브에서 **R 설치 동영상**으로 검색
- 직접 해보는 것이 가장 빠르게 배우는 길!

R Packages

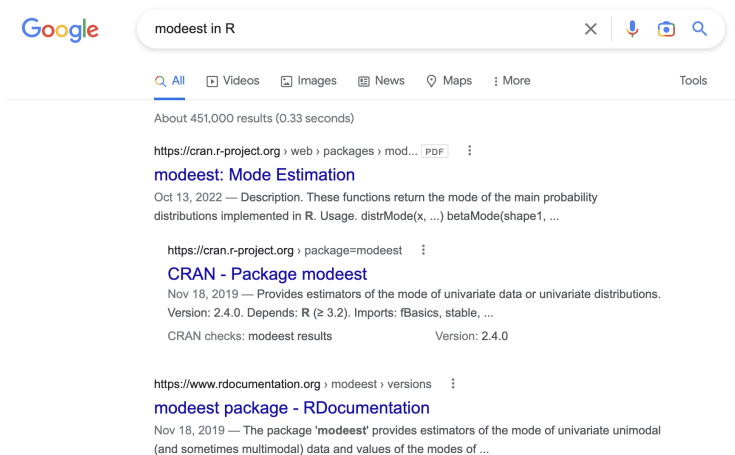
- 자료분석을 위한 간단한 통계적 방법에서부터 복잡한 방법론들이 대부분 R Packages 형태로 구현되어 있다.
- 새로운 방법을 시도해보기 전에 이미 구현된 패키지가 있는지 찾아보자.
- 예를들어 최빈값 (mode)를 찾는 함수를 구현해 놓은 modeest 패키지를 설치해 보자.
- R Studio의 오른쪽 아래 화면에서 “Packages” 탭을 선택하자. 이미 설치된 패키지의 목록을 볼 수 있을 것이다.

R Packages 설치하기



- ① “Install” 버튼을 클릭하자.
- ② 새로 뜨는 창에서 설치할 패키지의 이름을 입력하자. 첫 몇자만 입력하면 원하는 패키지가 보일 것이다. `modeest`를 전부 입력하거나 목록에서 선택하자.
- ③ “Install” 버튼을 누르면 R 콘솔에서 설치를 시작할 것이다.
- ④ 설치가 완료되면 패키지 창에 `modeest`가 추가될 것이다. 옆에 보이는 체크박스를 클릭하면 R 콘솔에서 `library(modeest)`가 실행되어 이 패키지를 로드한다.

- 설치된 패키지의 매뉴얼을 보는 방법이 있다.
- 가장 간단한 방법은 구글에서 “modeest in r” 라고 검색하는 것이다.



Google search results for "modeest in R".

Search bar: modeest in R

Results:

- About 451,000 results (0.33 seconds)

<https://cran.r-project.org> > web > packages > mod... PDF

modeest: Mode Estimation

Oct 13, 2022 — Description. These functions return the mode of the main probability distributions implemented in R. Usage. `distrMode(x, ...)` `betaMode(shape1, ...)`

<https://cran.r-project.org> > package=modeest

CRAN - Package modeest

Nov 18, 2019 — Provides estimators of the mode of univariate data or univariate distributions. Version: 2.4.0. Depends: R (≥ 3.2). Imports: fBasics, stable, ...

CRAN checks: modeest results Version: 2.4.0
- <https://www.rdocumentation.org> > modeest > versions

modeest package - RDocumentation

Nov 18, 2019 — The package 'modeest' provides estimators of the mode of univariate unimodal (and sometimes multimodal) data and values of the modes of ...

Package ‘modeest’

October 13, 2022

Type Package

Title Mode Estimation

Version 2.4.0

Description Provides estimators of the mode of univariate data or univariate distributions.

License GPL-3

LazyData TRUE

Depends R (>= 3.2)

Imports fBasics, stable, stabledist, stats, statip (>= 0.2.3)

Suggests evd, knitr, mvtnorm, testthat, VGAM

URL <https://github.com/paulponcet/modeest>

BugReports <https://github.com/paulponcet/modeest/issues>

RoxygenNote 7.0.0

NeedsCompilation no

Author Paul Poncet [aut, cre]

Maintainer Paul Poncet <paulponcet@yahoo.fr>

Repository CRAN

Date/Publication 2019-11-18 15:30:05 UTC

어떤 패키지가 있는지 어떻게 알까?

- Google 검색을 하거나 패키지 `ctv`를 이용하는 방법이 있다.
- 패키지 `ctv`를 설치해보자.
- `ctv` 를 이용해 몇 가지 주제 (베이지안, 계량경제학 등) 에 관련된 패키지를 한번에 다운 받을 수 있다.
- 한 가지 주제와 관련된 패키지들의 묶음을 “View”라고 한다
- cran.r-project.org/web/views/ 혹은 콘솔에서 `available.views()` 라고 실행하여 여러가지 주제의 “View” 목록을 볼 수 있다.

CRAN Task Views

CRAN task views aim to provide some guidance which packages on CRAN are relevant for tasks related to a certain topic. They give a brief overview of the included packages which can also be automatically installed using the [ctv](#) package. The views are intended to have a sharp focus so that it is sufficiently clear which packages should be included (or excluded) - and they are *not* meant to endorse the "best" packages for a given task.

To automatically install the views, the [ctv](#) package needs to be installed, e.g., via

```
install.packages("ctv")
```

and then the views can be installed via `install.views` or `update.views` (where the latter only installs those packages are not installed and up-to-date), e.g.,

```
ctv::install.views("Econometrics")
```

```
ctv::update.views("Econometrics")
```

To query information about a particular task view on CRAN from within R or to obtain the list of all task views available, respectively, the following commands are provided:

```
ctv::ctv("Econometrics")
```

```
ctv::available.views()
```

The resources available from the [CRAN Task View Initiative](#) provide further information on how to contribute to existing task views and how to propose new task views.

Topics

Agriculture	Agricultural Science
Bayesian	Bayesian Inference
CausalInference	Causal Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Databases	Databases with R
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
Epidemiology	Epidemiology

주제별 패키지 한번에 설치하기

- `ctv` 를 로딩한 후 Econometrics 관련 패키지를 설치하여 보자.
- `install.views("Econometrics")`
 - ▶ 위와 같이 입력하면 “Econometrics”에 포함된 패키지가 모두 설치된다.
- `install.views("Econometrics", coreOnly = TRUE)`
 - ▶ 위와 같이 입력하면 “Econometrics”에 포함된 패키지 중 core 패키지 (“AER” 외 8개)만 설치된다.

R 패키지 사용하기 - modeest

```
library(modeest)
```

```
mfv(c(1,2,1,2,3,3,3,4,5,4,5))
```

```
## [1] 3
```

```
mfv(c(1,5,7,7,9,9,10))
```

```
## [1] 7 9
```

R 기초

- R 을 실행하면 R 콘솔창이 제일먼저 보인다.
- 콘솔창에 직접 명령어를 입력할 수도 있지만 스크립트 창을 띄워 진행하는 것을 추천한다.
- 스크립트 창에 명령어를 작성하여 .R 파일로 저장할 수 있다.
(스크립트 창을 단지 텍스트 편집창으로 이용할 수 도 있다.)
- 스크립트 창을 이용하면 기존에 실행했던 명령어를 다시 타이핑할 필요 없이 재실행 가능하다.
- 스크립트 창에 작성된 명령어 한 줄을 실행하거나 선택된 영역을 실행하는 단축키는 *Ctrl + R* 또는 *Ctrl + enter* 이다.

- # 이후의 텍스트는 실행되지 않는다 (주석의 역할)

```
## nothing happen
```

- 간단한 계산 (계산기로 이용 가능)

```
3+5
```

```
## [1] 8
```

```
3/5
```

```
## [1] 0.6
```

- `help('+')` 명령어를 실행하게 되면 몇 가지 산술부호에 대한 설명을 볼 수 있다. (+, /, %%, %/% 등)

- 변수를 지정하여 계산을 할 수도 있다.

```
x = 5; y = 6  
x + y
```

```
## [1] 11
```

```
sin(x) + exp(y)
```

```
## [1] 402.4699
```

- “ ; ”를 이용해 한 줄에 여러개의 명령어를 작성하여 실행할 수 있다.

R 에서 다루는 자료형식

- 논리 (1 or 0) - TRUE or FALSE
`logical(length = 0), as.logical(x, ...),`
`is.logical(x)`
- 정수 (integers)
`integer(length = 0), as.integer(x, ...),`
`is.integer(x)`
- 부동 소수점수 (floating point numbers)
`double(length = 0), as.double(x, ...),`
`is.double(x), numeric(length = 0),`
`as.numeric(x, ...), is.numeric(x)`

- 문자열 (characters) `character(length=0)`, `as.character(x, ...)`,
`is.character(x)`
- NaN (Not a Number) `is.nan(x)`
- 양의 무한대, 음의 무한대 (Infinite, -Infinite)
`is.finite(x)`, `is.infinite(x)`
- NA (유효하지 않음, Not Available) / (결측값, Missing Values)
`is.na(x)`

예제 1

```
is.numeric(10)
```

```
## [1] TRUE
```

```
is.numeric('3')
```

```
## [1] FALSE
```

```
is.numeric("3")
```

```
## [1] FALSE
```

예제 2

```
is.character(5)
```

```
## [1] FALSE
```

```
is.character('5')
```

```
## [1] TRUE
```

```
is.character("5")
```

```
## [1] TRUE
```

```
is.infinite(-Inf)
```

```
## [1] TRUE
```

R에서 데이터를 저장하는 객체들

① 벡터 (Vector)

- 두 개 이상 나열된 값을 저장하는 방법으로, 값을 직접 할당하여 정의할 수 있다.

```
x1 <- c(1, 3, 5, 7, 9)
```

```
x1
```

```
## [1] 1 3 5 7 9
```

- 벡터는 R 기초 함수 및 명령어를 사용하여 정의할 수도 있다.

```
x2 = seq(1, 9, 2)
```

```
x2
```

```
## [1] 1 3 5 7 9
```

```
x3 = rep(1, 10)
```

```
x3
```

```
## [1] 1 1 1 1 1 1 1 1 1 1
```

```
x4 = 1:10
```

```
x4
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

- 벡터는 characters를 저장할 수도 있다.

```
x5 = c('1', '2', '3')  
x5
```

```
## [1] "1" "2" "3"
```

```
x6 = c('A', 'B', 'C')  
x6
```

```
## [1] "A" "B" "C"
```


- 벡터의 계산은 각각의 성분에 대해서 이루어진다. (element-wise)

```
x1 ^ 2 + 2 * x2 ^ 2
```

```
## [1] 3 27 75 147 243
```

- 벡터의 길이가 서로 다르지만 한 벡터의 길이가 다른 벡터의 길이의 배수인 경우 조심하자. 짧은 벡터가 그 배수만큼 재사용된다.

```
x1; x3
```

```
## [1] 1 3 5 7 9
```

```
## [1] 1 1 1 1 1 1 1 1 1 1
```

```
x1 + x3
```

```
## [1] 2 4 6 8 10 2 4 6 8 10
```

② 리스트 (List)

- 벡터는 동일한 자료형을 갖는 값들을 저장하지만 리스트는 서로 다른 자료형을 저장할 수 있다.
- 자료형 뿐만아니라 자료의 길이가 달라도 리스트로 저장이 가능하다.

```
myvector <- c(8, 6, 9, 10, 5)
mylist <- list(name = "Fred", wife = "Mary", myvector)
mylist
```

```
## $name
## [1] "Fred"
##
## $wife
## [1] "Mary"
##
## [[3]]
## [1] 8 6 9 10 5
```

- 리스트 내 각 개체는 번호나 이름으로 따로 불러올 수 있다.

```
mylist[[2]]
```

```
## [1] "Mary"
```

```
mylist$wife
```

```
## [1] "Mary"
```

```
mylist[[3]]
```

```
## [1] 8 6 9 10 5
```

③ 테이블 (table)

- 범주형 변수에 대한 분할표 (contingency table) 를 생성한다.

```
mynames <- c("Mary", "John", "Ann", "Sinead", "Joe",  
             "Mary", "Jim", "John", "Simon")  
table(mynames)
```

```
## mynames  
##      Ann      Jim      Joe      John      Mary      Simon      Sinead  
##       1       1       1       2       2       1       1
```

- 리스트와 마찬가지로 번호나 이름으로 각 개체를 불러올 수 있다.

```
mytable <- table(mynames)
mytable[[4]]
```

```
## [1] 2
```

```
mytable[["John"]]
```

```
## [1] 2
```

❶ 행렬 (Matrix)

- 자료를 행렬의 형태로 저장한다.

```
mymatrix <- matrix(1:12, nrow = 4, ncol = 3)
mymatrix
```

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

- 좁은 의미에서 행렬은 숫자만을 값으로 갖는다.

⑤ 배열 (Array)

- 행과 열 이상의 차원을 갖는 자료를 저장한다.

```
myarray <- array(1:8, dim = c(2, 2, 2))  
myarray
```

```
## , , 1
```

```
##
```

```
##      [,1] [,2]
```

```
## [1,]    1    3
```

```
## [2,]    2    4
```

```
##
```

```
## , , 2
```

```
##
```

```
##      [,1] [,2]
```

```
## [1,]    5    7
```

```
## [2,]    6    8
```

데이터 프레임 (Data Frame)

- 데이터를 나타낼 때 기본적으로면서도 많이 쓰이는 방법이 행과 열로 구성된 행렬 형식으로 나타내는 것이다.
- 각 행은 경우(case) 또는 사례(instance)를, 열은 변수(variable) 또는 속성(attribute)을 나타낸다.
- 대부분의 스프레드시트 프로그램에서 데이터를 이와 같이 취급하며, R에서는 이를 데이터 프레임(data frame)이라 부르는 대상(object)으로 나타낸다.

변수와 속성

- 한 가족에 대한 자료를 예로 들어보자.

Name	Age	Gender	Weight
Dad	43	Male	188
Mom	42	Female	136
Sis	12	Female	83
Bro	8	Male	61
Dog	5	Female	44

- 첫 행(row)은 변수(variable) 또는 속성(attribute)의 이름을 담고 있다.
- 예를 들어, age는 모든 생명체가 가지고 있는 속성이다.
- 이런 점에서, 표의 첫 행은 “메타데이터”, 즉 데이터에 대한 데이터라 부를 수 있다.

경우와 사례

Name	Age	Gender	Weight
Dad	43	Male	188
Mom	42	Female	136
Sis	12	Female	83
Bro	8	Male	61
Dog	5	Female	44

- 첫 행을 제외한 나머지 행들은 경우(case) 또는 사례(instance)를 나타낸다.
- 각 행은 한 가족에 속한 생명체(사람 혹은 개)를 나타낸다.

열(column)

- 한 열(column)은 위아래로 모두 동일한 종류의 값을 가지고 있다.
- 예를 들어, “Age” 열은 모두 숫자로 되어 있다. “Old”나 “Young” 따위의 문자열 값은 가지지 않는다.
- 모든 열은 같은 수의 엔트리를 가지고 있다.
- 따라서 데이터 프레임은 직사각형 형태를 지닌다.
- 데이터과학자는 다른 데이터가 직사각형 형태를 띄고 있기를 바란다.

데이터프레임 만들기

각 열을 하나씩 만들어 하나의 데이터프레임으로 합칠 수 있다.

```
myFamNames = c("Dad", "Mom", "Sis", "Bro", "Dog")  
myFamNames
```

```
## [1] "Dad" "Mom" "Sis" "Bro" "Dog"
```

```
myFamAges = c(43, 42, 12, 8, 5)  
myFamGenders = c("Male", "Female", "Female", "Male", "Female")  
myFamWeights = c(188, 136, 83, 61, 44)
```

각 열은 벡터로, 한가지 유형의 타입이어야 한다. 즉, 한 벡터의 원소는 모두 숫자이거나 모두 문자열이거나 해야 한다.

길이가 같은 벡터들을 모아 데이터프레임을 만들 수 있다.

```
myFamily <- data.frame(myFamNames, myFamAges,  
                        myFamGenders, myFamWeights)  
myFamily
```

##	myFamNames	myFamAges	myFamGenders	myFamWeights
## 1	Dad	43	Male	188
## 2	Mom	42	Female	136
## 3	Sis	12	Female	83
## 4	Bro	8	Male	61
## 5	Dog	5	Female	44

데이터 프레임 요약하기

데이터프레임의 구조와 내용을 간략하게 보고 싶을 때 `str()` 함수를 사용한다.

```
str(myFamily)
```

```
## 'data.frame':    5 obs. of  4 variables:
##  $ myFamNames   : chr  "Dad" "Mom" "Sis" "Bro" ...
##  $ myFamAges    : num  43 42 12 8 5
##  $ myFamGenders: chr  "Male" "Female" "Female" "Male" ...
##  $ myFamWeights: num  188 136 83 61 44
```

각 변수의 유형과 선두 사례 몇 개를 각 줄의 첫번째 쌍점(:) 오른쪽에 보여준다.

요인 (Factor)

- myFamGenders와 같은 범주형 변수는 R에서 요인(factor) 형태로 분석한다.
- 각 범주는 레벨(level)이라고 한다.
- 문자열변수를 요인형태으로 데이터프레임에 저장하기 위해서는 stringsAsFactors = TRUE 옵션을 준다.

```
myFamilyStr <- data.frame(myFamNames,  
                           myFamAges,  
                           myFamGenders,  
                           myFamWeights,  
                           stringsAsFactors = TRUE)
```

```
str(myFamilyStr)
```

```
## 'data.frame':    5 obs. of  4 variables:
## $ myFamNames   : Factor w/ 5 levels "Bro","Dad","Dog",...: 2 1 1 1 1
## $ myFamAges    : num  43 42 12 8 5
## $ myFamGenders: Factor w/ 2 levels "Female","Male": 2 1 1 1 1
## $ myFamWeights: num  188 136 83 61 44
```

- R은 내부적으로 각 범주의 이름을 알파벳 순으로 자연수 위에 덧붙인다. 즉, 성별 요인은 “Female”을 1로 “Male”을 2로 저장한다. 두번째 쌍점(:) 뒤의 숫자가 이를 나타낸다.
- 원래의 변수 myFamGenders는 문자열 벡터임을 주의하자.

데이터프레임 요약하기

```
summary(myFamily)
```

```
##      myFamNames      myFamAges myFamGenders      myFamV
## Length:5          Min.      : 5      Length:5          Min.
## Class :character  1st Qu.: 8      Class :character  1st Qu.
## Mode  :character  Median :12     Mode  :character  Median
##                      Mean  :22                      Mean
##                      3rd Qu.:42                      3rd Qu.
##                      Max.   :43                      Max.
```

summary()는 str()과 비슷한 결과를 보이나 각 변수형에 따라 적절한 요약 통계량을 출력한다.

데이터프레임 내의 변수접근

- 데이터프레임은 벡터로 이루어진 열들의 리스트(list)로 되어 있다.
- 리스트는 (키,값) 형태의 데이터를 담는 연관 배열(associative array)로, “값”은 “리스트명\$키” 형식으로 얻어낼 수 있다.
- myFamNames에서는 이 데이터프레임을 만들 때 사용한 벡터 변수 이름들이 키가 된다. 이를 이용해 특정 열의 값들을 얻어낼 수 있다.

```
myFamily$myFamAges
```

```
## [1] 43 42 12 8 5
```

- 특정 행을 얻어내기 위해서는 행번호를 사용하면된다.

```
myFamily[2,]
```

```
##      myFamNames myFamAges myFamGenders myFamWeights
## 2           Mom         42         Female         136
```

그냥 “myFamAges”만 쓰면 안될까?

```
myFamAges <- c(myFamAges, 11)
myFamAges
```

```
## [1] 43 42 12 8 5 11
```

```
myFamily$myFamAges
```

```
## [1] 43 42 12 8 5
```

- 데이터프레임의 myFamAges 과 이를 만들 때 사용했던 myFamAges 벡터는 서로 다른 환경에 존재하며 서로 무관하다.

데이터프레임의 한 열 끝에 값을 추가하면 어떻게 될까?

```
myFamily$myFamAges <- c(myFamily$myFamAges, 11)
```

```
## Error in `$<-.data.frame`(`*tmp*`, myFamAges, value = c(43,
```

- 생략된 문구는 “replacement has 6 rows, data has 5”.
- 즉, 데이터프레임의 모든 열은 길이가 같아야 함을 상기하자.

데이터 프레임 가지고 놀기

예제를 가지고 데이터프레임을 좀 더 깊게 다루는 방법들을 소개한다.

Iris 데이터

- 19세기말 ~ 20세기초의 유명 통계학자인 피셔(R. A. Fisher)가 연구한 데이터
- 붓꽃(iris)의 3가지 종(setosa, versicolor, virginica)에 대해 꽃받침(sepal)과 꽃잎(petal)의 길이를 정리한 데이터

- Species: 붓꽃의 종 (범주형 변수; Factor)
- Sepal.Width: 꽃받침의 너비 (수치형 변수; numeric)
- Sepal.Length: 꽃받침의 길이 (수치형 변수; numeric)
- Petal.Width: 꽃잎의 너비 (수치형 변수; numeric)
- Petal.Length: 꽃잎의 길이 (수치형 변수; numeric)

```
summary(iris)
```

```
##      Sepal.Length      Sepal.Width      Petal.Length      Petal.Width
## Min.      :4.300      Min.      :2.000      Min.      :1.000      Min.      :0.100
## 1st Qu.:5.100      1st Qu.:2.800      1st Qu.:1.600      1st Qu.:0.400
## Median :5.800      Median :3.000      Median :4.350      Median :1.300
## Mean    :5.843      Mean    :3.057      Mean    :3.758      Mean    :1.600
## 3rd Qu.:6.400      3rd Qu.:3.300      3rd Qu.:5.100      3rd Qu.:1.800
## Max.    :7.900      Max.    :4.400      Max.    :6.900      Max.    :2.500
##
##      Species
## setosa      :50
## versicolor:50
## virginica   :50
##
##
##
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 .
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.2
##  $ Species      : Factor w/ 3 levels "setosa","versicolor",.
```



```
head(iris)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5         1.4         0.2   setosa
## 2           4.9         3.0         1.4         0.2   setosa
## 3           4.7         3.2         1.3         0.2   setosa
## 4           4.6         3.1         1.5         0.2   setosa
## 5           5.0         3.6         1.4         0.2   setosa
## 6           5.4         3.9         1.7         0.4   setosa
```

apply 함수들

행렬, 리스트, 데이터프레임 등에 임의의 함수를 일괄적으로 적용한 결과를 얻기 위한 함수들: `apply`, `lapply`, `sapply`, `tapply`

- ① `apply(X, MARGIN, FUN)`
 - 행렬(X)의 각 행(MARGIN = 1) 또는 열(MARGIN=2) 방향으로 특정 함수(FUN)를 일괄 적용
 - e.g., 각 열의 합 구하기

```
apply(iris[, 1:4], 2, sum)
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width  
##           876.5           458.6           563.7           179.9
```

② lapply(X, FUN) # list apply

- 리스트(x)의 각 벡터에 특정 함수(FUN) 일괄 적용. 결과값은 리스트.
- e.g., 각 벡터(열)의 평균 구하기

```
lapply(iris[, 1:4], mean)
```

```
## $Sepal.Length  
## [1] 5.843333  
##  
## $Sepal.Width  
## [1] 3.057333  
##  
## $Petal.Length  
## [1] 3.758  
##  
## $Petal.Width  
## [1] 1.199333
```

③ `sapply(X, FUN)` # simplified `lapply`

- `lapply()`와 비슷하지만 결과값은 리스트 혹은 행렬일 수 있음.
- 리스트의 각 벡터에 적용된 함수 결과값이 서로 같은 길이를 가질 때 행렬로 요약하여 결과를 출력함.

```
sapply(iris[, 1:4], mean)
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width  
##      5.843333      3.057333      3.758000      1.199333
```

- 함수부분에 직접 함수를 정의할 수도 있다.

```
y = sapply(iris[, 1:4], function(x){x > 3})  
head(y, 3)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width  
## [1,]          TRUE          TRUE          FALSE          FALSE  
## [2,]          TRUE          FALSE          FALSE          FALSE  
## [3,]          TRUE          TRUE          FALSE          FALSE
```

❶ tapply(X, INDEX, FUN)

- 자료(X)들이 속한 그룹(INDEX) 별로 함수(FUN)를 적용
- INDEX: 데이터가 어느 그룹에 속하는지를 표현하기 위한 요인 변수
- e.g., Species별 Sepal.Length의 평균

```
levels(iris$Species)
```

```
## [1] "setosa"      "versicolor" "virginica"
```

```
tapply(iris$Sepal.Length, iris$Species, mean)
```

```
##      setosa versicolor  virginica  
##      5.006      5.936      6.588
```

데이터 분리 함수들 (split, subset)

① split(x, f)

- 데이터(x)를 각 요인(f)에 따라 분리해주는 함수

```
byspecies <- split(iris, iris$Species)  
str(byspecies)
```

```
## List of 3
```

```
## $ setosa      : 'data.frame':  50 obs. of  5 variables:
```

```
## ..$ Sepal.Length: num [1:50] 5.1 4.9 4.7 4.6 5 5.4 4.6 5
```

```
## ..$ Sepal.Width : num [1:50] 3.5 3 3.2 3.1 3.6 3.9 3.4 3
```

```
## ..$ Petal.Length: num [1:50] 1.4 1.4 1.3 1.5 1.4 1.7 1.4
```

```
## ..$ Petal.Width : num [1:50] 0.2 0.2 0.2 0.2 0.2 0.4 0.3
```

```
## ..$ Species      : Factor w/ 3 levels "setosa","versicolor"
```

```
## $ versicolor: 'data.frame':  50 obs. of  5 variables:
```

```
## ..$ Sepal.Length: num [1:50] 7 6.4 6.9 5.5 6.5 5.7 6.3 4
```

```
## ..$ Sepal.Width : num [1:50] 3.2 3.2 3.1 2.3 2.8 2.8 3.3
```

```
## ..$ Petal.Length: num [1:50] 4.7 4.5 4.9 4 4.6 4.5 4.7 3
```

```
## ..$ Petal.Width : num [1:50] 1.4 1.5 1.5 1.3 1.5 1.3 1.6
```

② subset(x, subset)

- 데이터(x)에서 특정 조건(subset)을 만족하는 사례/행을 분리해주는 함수

```
setosa <- subset(iris, Species == "setosa")  
head(setosa)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.1	3.5	1.4	0.2	setosa
## 2	4.9	3.0	1.4	0.2	setosa
## 3	4.7	3.2	1.3	0.2	setosa
## 4	4.6	3.1	1.5	0.2	setosa
## 5	5.0	3.6	1.4	0.2	setosa
## 6	5.4	3.9	1.7	0.4	setosa

② subset(x, select)

- 데이터(x)에서 특정 조건(select)을 만족하는 변수/열만 분리하기 위해서도 사용 가능.

```
colsel <- subset(iris, select = c(Sepal.Length, Species))  
head(colsel)
```

##	Sepal.Length	Species
## 1	5.1	setosa
## 2	4.9	setosa
## 3	4.7	setosa
## 4	4.6	setosa
## 5	5.0	setosa
## 6	5.4	setosa

② subset(x, select)

- 데이터(x)에서 특정 조건(select)을 만족하는 변수/열을 제외하기 위해서도 사용 가능.

```
colsel <- subset(iris, select = -c(Sepal.Length, Species))  
head(colsel)
```

##	Sepal.Width	Petal.Length	Petal.Width
## 1	3.5	1.4	0.2
## 2	3.0	1.4	0.2
## 3	3.2	1.3	0.2
## 4	3.1	1.5	0.2
## 5	3.6	1.4	0.2
## 6	3.9	1.7	0.4

데이터 정렬하기

```
x <- c(20, 11, 33, 50, 47)
sort(x)
```

```
## [1] 11 20 33 47 50
```

```
order(x)
```

```
## [1] 2 1 3 5 4
```

```
x[order(x)]
```

```
## [1] 11 20 33 47 50
```

- iris를 Sepal.Length에 따라 정렬

```
iris.ordered <- iris[order(iris$Sepal.Length), ]  
head(iris.ordered)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 14	4.3	3.0	1.1	0.1	setosa
## 9	4.4	2.9	1.4	0.2	setosa
## 39	4.4	3.0	1.3	0.2	setosa
## 43	4.4	3.2	1.3	0.2	setosa
## 42	4.5	2.3	1.3	0.3	setosa
## 4	4.6	3.1	1.5	0.2	setosa

- Sepal.Length가 동차일 경우 Petal.Length의 순서에 따라 정렬

```
head(iris[order(iris$Sepal.Length , iris$Petal.Length), ])
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 14	4.3	3.0	1.1	0.1	setosa
## 39	4.4	3.0	1.3	0.2	setosa
## 43	4.4	3.2	1.3	0.2	setosa
## 9	4.4	2.9	1.4	0.2	setosa
## 42	4.5	2.3	1.3	0.3	setosa
## 23	4.6	3.6	1.0	0.2	setosa

데이터 섞기

```
sample(1:10, 5) # 비복원추출 (중복 불허)
```

```
## [1] 4 3 8 9 1
```

```
sample(1:10, 5, replace = TRUE) # 복원추출 (중복 허용)
```

```
## [1] 10 9 1 3 7
```

```
iris.resample <- iris[sample(NROW(iris), NROW(iris)), ]  
head(iris.resample, 5)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width      Species  
## 143           5.8         2.7           5.1          1.9 virginica  
## 119           7.7         2.6           6.9          2.3 virginica  
## 104           6.3         2.9           5.6          1.8 virginica  
## 12            4.8         3.4           1.6          0.2 setosa  
## 81            5.5         2.4           3.8          1.1 versicol
```

그룹별 연산 함수 (aggregate)

① `aggregate(x, by, FUN)`

- 데이터(x)를 각 그룹(by) 별로 함수(FUN) 적용 (c.f., `tapply`)

② `aggregate(y ~ x, data, FUN)`

- 데이터(data)의 변수 y를 그룹 변수(x) 별로 함수(FUN) 적용
- `y ~ x`: 처리할 변수들을 일종의 수식(formula)으로 표현하는 방법
- e.g., 종별 Sepal.Width의 평균 길이 구하기

```
aggregate(Sepal.Width ~ Species, iris, mean)
```

```
##      Species Sepal.Width
## 1      setosa      3.428
## 2 versicolor      2.770
## 3  virginica      2.974
```

시행착오에서 배우기

- R 은 기본적으로 프로그래밍 **언어**이므로 교재의 명령어들을 직접 실행해보아야 효율적으로 학습할 수 있다.
- 명령어들이 제대로 실행되지 않을 경우 기본적으로 검토해 볼 사항:
 - ▶ 스펠링, 구두점, 대소문자 확인
 - ▶ 자료의 변수형, 결측값, 차원 확인
 - ▶ 온라인에 Error or Warning 메시지 검색
- 배운 것을 조합하여 새로운 시도를 해 볼 필요가 있음.

정리

- R은 다양한 통계 기법과 그래픽 능력을 갖춘 오픈 소스 데이터분석 소프트웨어이다.
- 오픈 소스의 특성상, 새로운 기능에 대한 필요가 생기면 그 구현이 대단히 빠르며, 이는 사용자 수 증가에 큰 요인이 되었다.
- R은 다양한 컴퓨터 환경을 지원하며, 설치가 매우 쉽다.
- R은 기본적으로 프로그래밍 언어이며, 명령어 기반으로 작동한다.
- 통합개발환경(IDE)인 R Studio를 사용하여 새로운 함수 작성 및 패키지 관리를 쉽게 할 수 있다.
- 패키지는 추가적인 함수들의 집합이며 R커뮤니티의 개발자들이 오픈소스로 공유하고 있다.

참고자료

- Avril Coghlan, A Little Book of R for Biomedical Statistics
<http://a-little-book-of-r-for-biomedical-statistics.readthedocs.org/>
- Jeffrey Stanton, Introduction to Data Science
<http://jsresearch.net/wiki/projects/teachdatascience>
- Quick R <http://www.statmethods.net/>
- Stack Overflow <http://stackoverflow.com/questions/tagged/r>