제1강 : R의 소개

금융 통계 및 시계열 분석

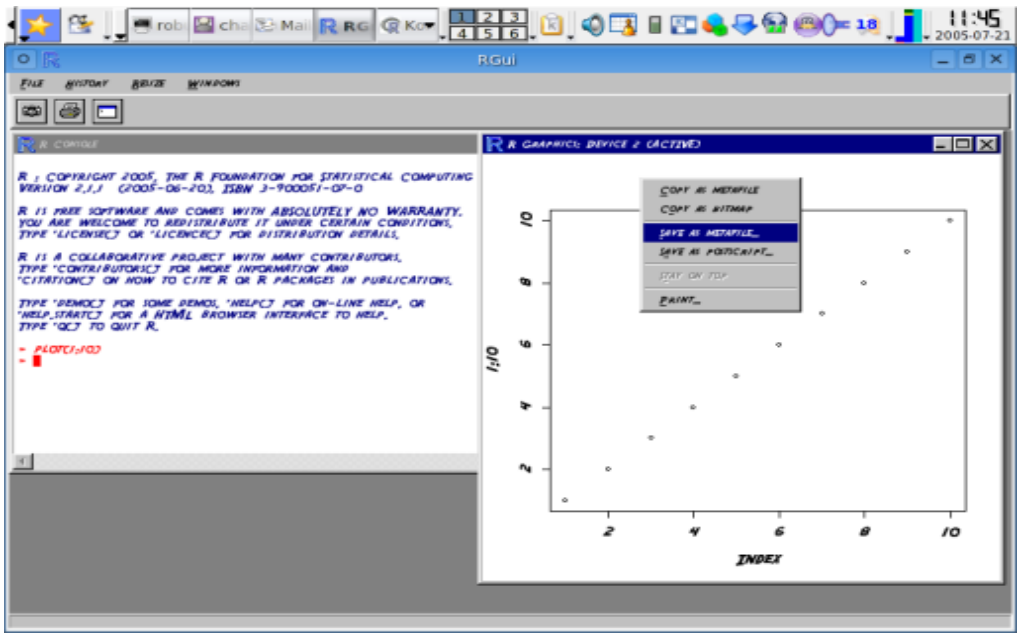
TRADE INFORMATIX

2013년 11월 11일

Outline

R이란?

- S/S+ 언어의 무료 오픈소스 구현 (GPL)
  - S: AT&T 벨 연구소에서 개발한 통계용 프로그래밍 언어
  - S+: TIBCO Software에서 개발한 S 프로그래밍 언어의 상용 버전
- 통계계산용 프로그래밍 언어와 환경 (environment)
- 언어 정의와 쉘 기능, 그래픽스 출력 기능 통합
- 강력한 패키지 개발 지원 기능, 2013년 11월 현재 5000개 이상의 패키지 제공
- 윈도우즈, 맥킨토시, 리눅스 등 다양한 플랫폼에서 사용가능

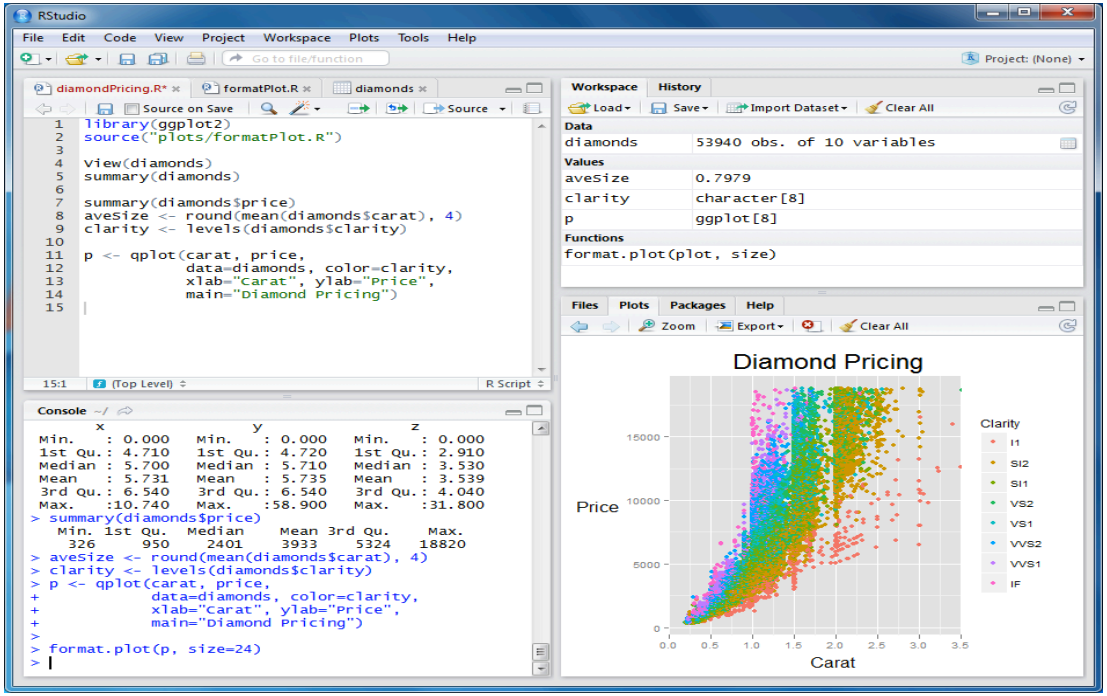


R 참고 자료

- web
  - `http://www.R-project.org` R 프로젝트 홈페이지. R 프로그램. 패키지 다운로드
  - `http://cran.r-project.org/manuals.html` R 영문 매뉴얼
  - `http://r-project.kr/` 한국 R 사용자 모임
  - `http://ihelp.r-forge.r-project.org//lang.html` R 도큐멘트 한글화 작업
- 도서
  - 빅데이터 분석 도구 R 프로그래밍, 노만 매트로프 지음, 권정민 옮김
  - R Cookbook, 폴 티터 지음, 이제원 옮김
  - RStudio 따라잡기, 마크 P. J. 판 데르 루, 에드윈 데 용에 지음, 정사범 옮김
  - 등 한글 도서 30여종 출판

RStudio

- 무료 오픈소스 통합개발환경 (IDE: Integrated Development Environment)
- R 콘솔, bash 콘솔, 에디터, 워크스페이스 뷰, 히스토리, 파일 뷰, 패키지 뷰, 헬프 뷰, 플롯 뷰 등 제공
- svn, git 소스 콘트롤 시스템 통합
- 데스크탑 버전 및 서버 버전 제공



RStudio Server

- ❏ 리눅스 서버에 설치
- ❏ 리눅스 서버의 사용자 계정과 연동
- ❏ 웹브라우저를 통해 데스크탑 버전과 동일한 GUI 사용
- ❏ 사용자는 R 패키지나 데이터의 설치
- ❏ 서버에 설치된 패키지 및 데이터베이스의 공동 이용 가능

R 설치

- `http://CRAN.R-project.org`에서 다운로드
- 윈도우즈 버전은 `http://cran.r-project.org/bin/windows/base/`에서 다운로드
- 윈도우즈 버전은 더블클릭으로 간단설치

R 가동과 종료

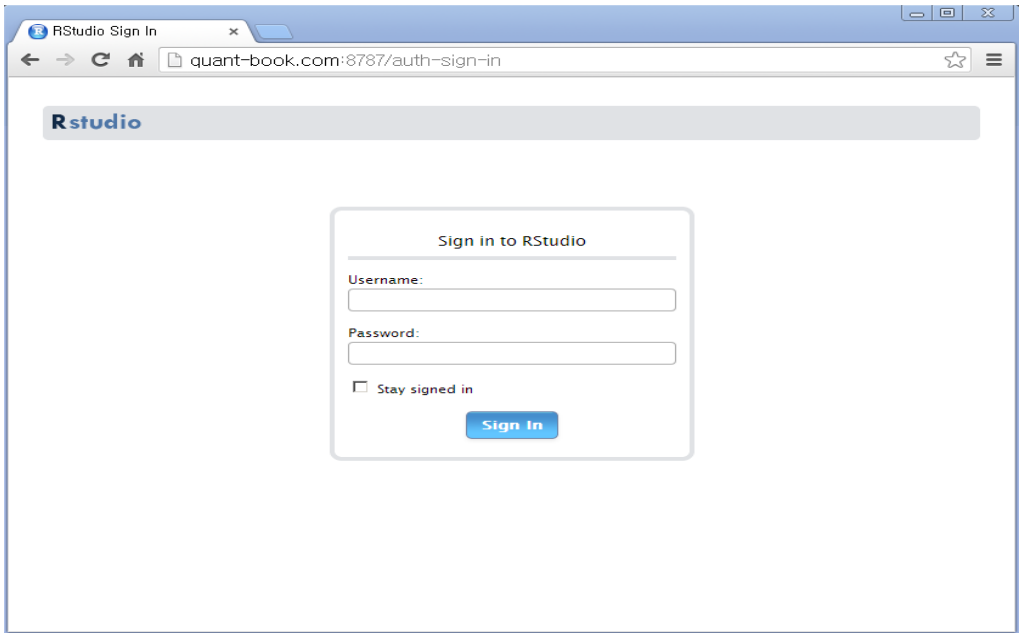
- 가동
  - R 아이콘 더블클릭
  - 콘솔창에서 `R` 명령
- 종료
  - R 콘솔에서 `q()` 명령
  - 동작중인 명령의 비상 종료: `ctrl + c`
  - R 프로그램의 비상 종료: `ctrl + d`
  - 콘솔화면 지우기 : `ctrl + L`
  - 종료시에는 워크스페이스를 저장할지 물어봄 (y 이면 저장)

R 가동과 종료 실습

```
R version 2.15.2 (2012-10-26) -- "Trick or Treat"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> q()
Save workspace image? [y/n/c]: y
#
```

Quant-Book RStudio 로그인

- 웹브라우저에서 주소 `http://quant-book.com:8787`
- 로그인 아이디와 패스워드 입력



R 도움말 사용법

- 공식 매뉴얼 `http://cran.r-project.org/manuals.html`
- `help.start()` : 도움말 홈 화면
- `help.search()` : 도움말 검색
- `help()` : 특정 명령어에 대한 도움말
  - 명령문 앞에 `?` 붙이기와 동일

```
> help.start()
> help.search("plot")
> help("plot")
> ?plot
```

R의 계산기능

- 계산 기능

```
> sqrt(25) + 2
[1] 7
```

- 변수(오브젝트)에 값 대입 : = 기호와 <-, -> 기호 모두 사용 가능

```
> x <- 25
> y <- sqrt(x) + 2
```

- 변수(오브젝트)에 있는 값 조회

```
> print(x)
[1] 25
> y
[1] 7
```

R의 기본 연산자

기호	의미	기호	의미
`x + y`	덧셈	`x & y`	논리 AND
`x - y`	뺄셈	`x | y`	논리 OR
`x * y`	곱셈	`!x`	논리 NOT
`x / y`	실수 나눗셈	`x == y`	같음
`x ^ y`	거듭제곱	`x != y`	다음
`x %% y`	나머지	`x <= y`	작거나 같음
`x %/% y`	정수 나눗셈	`x >= y`	크거나 같음

- 산술 연산 http://stat.ethz.ch/R-manual/R-devel/library/base/html/Arithmetic.html
- 삼각 함수 http://stat.ethz.ch/R-manual/R-devel/library/base/html/Trig.html
- 쌍곡선 함수 http://stat.ethz.ch/R-manual/R-devel/library/base/html/Hyperbolic.html
- 로그/지수 함수http://stat.ethz.ch/R-manual/R-devel/library/base/html/Log.html
- 논리 함수 http://stat.ethz.ch/R-manual/R-devel/library/base/html/Logic.html
- 기타 함수http://stat.ethz.ch/R-manual/R-devel/library/base/html/MathFun.html

R 오브젝트

- 오브젝트
  - R에서 데이터를 저장하는 변수
  - 워크스페이스에(Workspace)에 저장됨
- 오브젝트 이름 규칙
  - 영문 대소문자, 숫자 사용
  - 대소문자 구분
  - R 키워드 사용 불가
  - .(period 마침표) 사용 가능

R 워크스페이스

- 워크스페이스 : 데이터 (오브젝트) 가 저장되는 공간
- `ls()` : 저장된 변수의 목록보기
- `rm()` : 저장된 변수를 삭제하기
- `rm(list=ls())` : 모든 변수를 삭제하기

```
> ls()
[1] "x" "y"
> rm(y)
> ls()
[1] "x"
> rm(list=ls())
> ls()
character(0)
```

워크스페이스를 파일로 저장하고 불러오기

- `save(list, file)` 워크스페이스 중 원하는 오브젝트들을 원하는 이름의 파일에 저장
- `load(file)` 파일에 저장된 오브젝트를 워크스페이스로 로딩
- `save.image()` 워크스페이스 전체를 `.RData` 파일에 저장

```
> x <- 1; y <- 2; z <- 3
> save(x, y, file="mydata")
> rm(list=ls())
> ls()

character(0)

> load("mydata")
> ls()

[1] "x" "y"
```

디렉토리 명령

- ❏ `getwd()` 현재 디렉토리
- ❏ `setwd(dirname)` 디렉토리 옮기기
- ❏ `dir.create(dirname)` 디렉토리 생성
- ❏ `unlink(dirname, recursive = TRUE)` 디렉토리 삭제

```
> getwd()
[1] "/home/user"
> dir()
character(0)
> dir.create("mydir")
> dir()
[1] "mydir"
> setwd("mydir")
> getwd()
[1] "/home/user/mydir"
> setwd("..")
> getwd()
[1] "/home/user"
> unlink("mydir", recursive = TRUE)
> dir()
character(0)
```

R 명령어 히스토리

- `history()` : 과거 명령어 목록 보기
- `savehistory(filename)` : 히스토리를 파일에 저장
- `loadhistory(filename))` : 파일에 저장된 히스토리 불러오기

```
> history()
> savehistory("myhistory")
> loadhistory("myhistory")
```

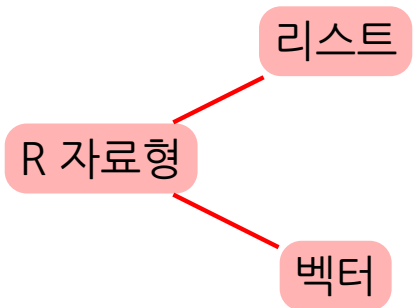
R 자료형의 분류

- R은 제너럴 프로그래밍 언어이기 때문에 자료형 (data type) 이 존재
- 그러나 R 오브젝트의 자료형은 다른 언어와 달리 분류 방법이 단일하지 않고 복잡함
- R 오브젝트의 자료형 분류 방법
  - "일반적인 카테고리 분류" 에 따른 자료형. 가장 널리 쓰임
  - `typeof` 명령으로 파악되는 자료형.
  - `mode` 명령으로 파악되는 자료형.
  - `storage.mode` 명령으로 파악되는 자료형.

R 자료형의 “일반적인 분류” - 기본

- ❏ R의 자료형을 ”수학적 개념”에 따라 묶거나 분리해 놓은 개념
- ❏ typeof 자료형과 1대1 일치하지 않음
- ❏ “일반적인 분류”가 같으면 사용방법이 유사하므로 익히기에 편리함

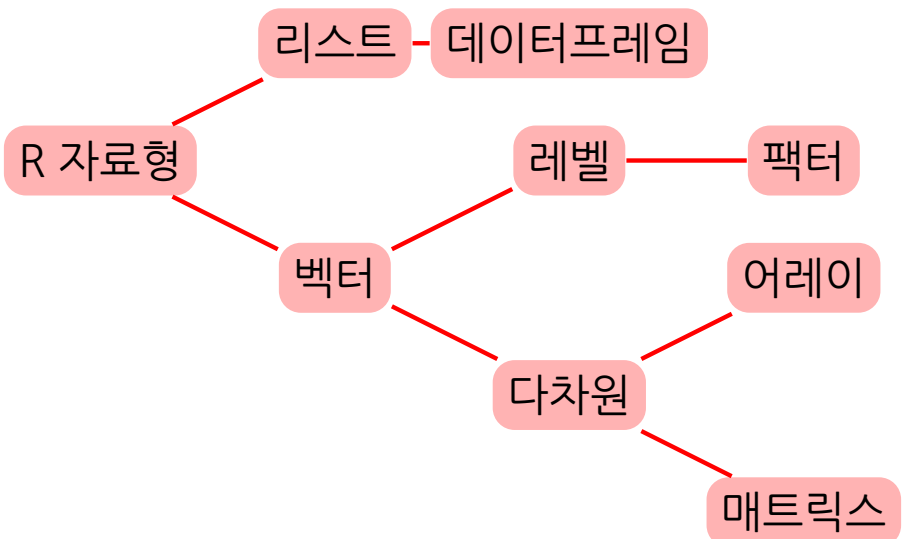
이름	의미	typeof
벡터 (vector)	동일 자료형 원소의 1차원 행렬	기본 벡터 자료형
리스트 (list)	다른 자료형 원소의 1차원 행렬	리스트



R 자료형의 “일반적인 카테고리 분류” - 고급

- ❏ 메트릭스 (2차원)/어레이 (다차원) 는 차원 (dim) 을 가지는 벡터
- ❏ 팩터는 레벨 (level) 을 가지는 벡터
- ❏ 데이터 프레임은 길이가 같은 벡터를 원소로 가지는 리스트

이름	의미	typeof
매트릭스 (matrix)	동일 자료형 원소의 2차원 행렬	기본 벡터 자료형
어레이 (array)	동일 자료형 원소의 n차원 행렬	기본 벡터 자료형
팩터 (factor)	level을 가지는 정수형 1차원 행렬	정수형 벡터 자료형
데이터프레임 (data.frame)	다른 자료형 원소의 2차원 행렬	벡터의 리스트



typeof 자료형

❑ typeof 명령어로 파악되는 자료형

```
> x <- "R"; y <- 3.14; z <- TRUE
> typeof(x); typeof(y); typeof(z)
[1] "character"
[1] "double"
[1] "logical"
```

typeof	의미	typeof	의미
"character"	문자열 벡터	"list"	리스트
"logical"	논리값 벡터	"symbol"	심볼
"integer"	정수 벡터	"closure"	함수
"double"	실수 벡터	"expression"	표현식
"complex"	복소수 벡터	"environment"	환경
"raw"	바이트 벡터	"S4"	S4 클래스
"NULL"	널 (null)	"..."	가변인수

“일반적인 분류”와 typeof 비교

- typeof 분류 방법에는 벡터, 매트릭스, 어레이, 데이터프레임이라는 자료형이 존재하지 않음.
- 벡터 자료형은 logical vector, integer vector 등의 6가지 기본(atomic) 벡터를 합쳐서 일컫는 용어.
- 팩터 자료형은 레벨(level) 속성(attribute)를 가지고 클래스(class) 속성이 `"factor"`인 정수형 벡터
- 매트릭스와 어레이는 벡터의 일종. 다만 차원(dimension) 속성(attribute)이 다른 벡터.
- 데이터프레임은 길이가 같은 벡터를 원소로 가지는 리스트. 클래스(class) 속성이 `"data.frame"`으로 명시.

R 벡터

- R의 가장 기본적인 자료형
- R에는 스칼라 자료형이 존재하지 않음. 모든 스칼라 값은 길이가 1인 벡터.
- 기본 벡터 : 일반적인 벡터 자료형에 해당하는 6가지 typeof 자료형
  - `"character"`
  - `"logical"`
  - `"integer"`
  - `"double"`
  - `"complex"`
  - `"raw"`

R 벡터의 생성 (creation)

- `":"` 1씩 증가/감소하는 순차적인 벡터 생성
- `seq(from, to, by)` 순차적인 벡터 생성
- `c()` 일반적인 벡터의 붙이기 (concatenation)
- `rep()` 반복
- `paste()`

R 벡터의 생성 (creation) 예 1

- ❏ `from:to` 에서 from이나 to가 정수가 아니라도 상관없음
- ❏ `from:to` 에서 to가 from과 정수차이가 아니면 내림 (증가시) 혹은 올림 (감소시)

```
> 1:4
[1] 1 2 3 4
> pi:6
[1] 3.141593 4.141593 5.141593
> 6:pi
[1] 6 5 4
> seq(17) # same as 1:17
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
> seq(1, 9, by = 2)
[1] 1 3 5 7 9
> seq(1, 6, by = 3)
[1] 1 4
> seq(1, 9, by = pi)
[1] 1.000000 4.141593 7.283185
```

R 벡터의 생성 예 2

- c() 명령은 임의의 벡터를 합칠수 있다.
- 합치는 벡터의 자료형이 다르면 가장 일반적인 자료형의 벡터가 된다.

```
> c(1,7:9)
[1] 1 7 8 9
> c("X1", "X2", "X3")
[1] "X1" "X2" "X3"
> typeof(1:3)
[1] "integer"
> c(1:3, 3.14)
[1] 1.00 2.00 3.00 3.14
> typeof(c(1:3, 3.14))
[1] "double"
> c("james", 3)
[1] "james" "3"
> typeof(c("james", 3))
[1] "character"
```

R 벡터의 생성 예 3

- `rep(x, times, each, length.out)` 명령은 x 벡터를 times 만큼 반복
- `times`가 길이 2 이상이면 각각의 원소를 times 원소만큼 반복
- `each`가 지정되면 모든 원소가 각각 `each`만큼 반복
- `times`와 `each`가 모두 지정되면 `each`만큼 각각의 원소가 반복된 벡터가 다시 `times` 만큼 반복
- `length.out`이 지정되면 그만큼만 출력

```
> rep(1:4, 2)
[1] 1 2 3 4 1 2 3 4
> rep(1:4, c(3,2,2,1))
[1] 1 1 1 2 2 3 3 4
> rep(1:4, c(2,2,2,2))
[1] 1 1 2 2 3 3 4 4
> rep(1:4, each = 2)
[1] 1 1 2 2 3 3 4 4
> rep(1:4, each = 2, len = 4)
[1] 1 1 2 2
> rep(1:4, each = 2, times = 3)
 [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

R 벡터의 생성 예 4

- paste(..., sep, collapse) 명령은 문자열을 붙이는 명령
- 입력 벡터가 문자열이 아니면 문자열로 변환
- 입력 벡터의 길이가 2이상이면 각각의 원소에 대해 붙임. 이때 두 개 이상의 입력 벡터의 길이가 서로 다르면 적은 벡터를 반복함
- sep은 각 원소를 붙일때 사이에 들어가는 문자. 디폴트는 sep=" "
- collapse 값이 지정되면 이를 이용하여 출력 벡터의 원소를 다시 합침

```
> paste(1:12)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11"
[12] "12"
> paste(1:12, collapse="+")
[1] "1+2+3+4+5+6+7+8+9+10+11+12"
> paste("a", 1:3)
[1] "a 1" "a 2" "a 3"
> paste(c("a", "b"), 1)
[1] "a 1" "b 1"
> paste(c("a", "b"), 1:5)
[1] "a 1" "b 2" "a 3" "b 4" "a 5"
> paste(c("a", "b"), 1:4, c("x", "y", "z"))
[1] "a 1 x" "b 2 y" "a 3 z" "b 4 x"
```

R의 인덱싱 (Indexing)

- 인덱싱은 벡터/리스트 등의 오브젝트의 특정 원소 혹은 특정 원소집합을 접근가능
- 벡터 인덱싱은 [] 기호 [[]]혹은 로 접근가능
- [] 기호 [[]] 안에는 정수벡터, 논리벡터, 이름 (문자열 벡터) 이 들어갈 수 있음
- 리스트 인덱싱의 경우에는 추가적으로 $ 기호를 사용 가능

R 벡터의 정수 벡터 인덱싱

- [] 안에 길이 1인 정수 벡터 n이 있으면 n번째 원소를 가리킴
- [] 안에 길이 2 이상인 정수 벡터가 있으면 그 정수 해당하는 원소들만 잘라냄 (slicing)
- 인덱스가 음수이면 반전 선택. 즉 그 인덱스를 제외한 나머지 원소 인덱싱
- 정수를 이용한 벡터 인덱싱의 경우에는 []와 [[]]이 동일 기능

```
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x[1]
[1] 1
> x[2:4]
[1] 2 3 4
> x[c(3,5,6,9)]
[1] 3 5 6 9
> x[-1:-2]
[1]  3  4  5  6  7  8  9 10
> x[[1]]
[1] 1
```

R 벡터의 논리 벡터 인덱싱

- 길이가 같은 논리 벡터를 이용한 인덱싱도 가능
- 논리 벡터 인덱스 길이가 벡터 길이보다 작으면 없는 인덱스는 TRUE로 가정
- 논리 벡터 인덱스 길이가 벡터 길이보다 길면 TRUE의 경우 NA원소 추가(append), FALSE의 경우 무시
- 벡터 원소중 특정 조건을 만족하는 원소를 뽑아내는 질의(query)에 유용

```
> x <- 1:10
> x[c(T,T,T,T,F,F,F,F,T,F)] # same length
[1] 1 2 3 4 9
> x[c(T,T,T,T,F,F,F,F,T)] # shorter
[1]  1  2  3  4  9 10
> x[c(T,T,T,T,F,F,F,F,T,F,T)] # longer with T
[1]  1  2  3  4  9 NA
> x[c(T,T,T,T,F,F,F,F,T,F,F)] # longer with F
[1] 1 2 3 4 9
> x[x>5]
[1]  6  7  8  9 10
> x[(x>5)&(x%%2==0)]
[1]  6  8 10
```

R 벡터의 이름 (names)

- 벡터를 포함한 모든 R 오브젝트는 이름 (names) 속성을 설정하여 각각의 원소에 이름을 붙일 수 있음.
- 이름 속성의 길이가 벡터 길이보다 길면 에러발생. 작으면 나머지 이름은 `NA`
- 이름을 이용한 인덱싱 가능

```
> x <- 1:10
> names(x) <- paste("x", 1:10, sep="")
> x
 x1  x2  x3  x4  x5  x6  x7  x8  x9 x10
  1   2   3   4   5   6   7   8   9  10
> names(x)
 [1] "x1"  "x2"  "x3"  "x4"  "x5"  "x6"  "x7"  "x8"  "x9"
[10] "x10"
> names(x) <- c("x1", "x2")
> x
  x1   x2 <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
   1    2    3    4    5    6    7    8    9   10
```

R 벡터의 이름(문자열) 벡터 인덱싱

- 이름이 있는 경우 문자열 벡터 인덱싱 가능
- [[]] 인덱싱은 이름 리턴값에서 이름 속성을 제거

```
> x <- 1:10
> names(x) <- paste("x", 1:10, sep="")
> x["x1"]
x1
 1
> x[c("x1", "x3")]
x1 x3
 1  3
> x[[1]]
[1] 1
> x[["x1"]]
[1] 1
```

R 벡터의 원소 추가 및 삭제

- 원칙적으로 R 벡터의 길이를 바꾸는 것은 불가능
- c() 명령어와 인덱스를 사용하여 재생성

```
> x <- 1:10
> y <- x[c(1:5, 7)]
> y
[1] 1 2 3 4 5 7
> z <- c(x, 11)
> z
 [1]  1  2  3  4  5  6  7  8  9 10 11
```

R 벡터의 속성 (attribute)

- NULL을 제외한 R 오브젝트는 속성 (attribute) 을 가질 수 있음
- names는 속성 중의 하나
- attributes() 명령으로 R 오브젝트가 가진 속성들 모두 나열

```
> x <- 1:10
> attributes(x)

NULL

> names(x) <- paste("x", 1:10, sep="")
> x

 x1  x2  x3  x4  x5  x6  x7  x8  x9 x10
  1   2   3   4   5   6   7   8   9  10

> attributes(x)

$names
 [1] "x1"  "x2"  "x3"  "x4"  "x5"  "x6"  "x7"  "x8"  "x9"
[10] "x10"

> attributes(x) <- NULL
> x

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

> attributes(x)

NULL
```

R 벡터의 속성 (계속)

❏ `attr(x, attrname)` 명령으로 R 오브젝트의 개별 속성을 설정하거나 볼 수 있음

```
> x <- 1:10
> attributes(x)

NULL

> attr(x, "myattr")

NULL

> attr(x, "myattr") <- "myattr_value"
> attr(x, "myattr")

[1] "myattr_value"

> attributes(x)

$myattr
[1] "myattr_value"
```

R 벡터의 특별 속성 (special attributes)

- R 오브젝트는 4가지의 특별 속성을 가짐
  - `names` 원소의 이름
  - `dim` 차원 (매트릭스, 어레이 정의에 사용)
  - `dimnames` 차원 인덱싱을 위한 각 차원의 이름
  - `class` OOP를 위한 클래스 속성

R 벡터의 차원 (dim)

❑ R 벡터는 `dim` 속성에 의해 `class` 속성이 `matrix/array` 로 자동 설정

```
> x <- 1:12
> dim(x)

NULL

> dim(x) <- c(3,4)
> x

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

> class(x)

[1] "matrix"

> dim(x) <- c(2,3,2)
> x

, , 1

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

, , 2

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

> class(x)

[1] "array"
```

R 매트릭스

- `matrix(data, nrow, ncol, byrow=TRUE)` : 매트릭스 생성
- `data, nrow, ncol`는 각각 초기화 데이터, 행수, 열수
- `byrow=TRUE`이면 행순서로 초기화, `byrow=FALSE`이면 열순서로 초기화
- `[row,col]` 방식으로 인덱싱. row/col 값이 생략되면 그 행/열 전체를 가리킴

```
> x <- matrix(nrow=2, ncol=3)
> x

     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA

> x[1,1] <- 1; x[1,2] <- 2; x[1,3] <- 3; x[2,1] <- 4; x[2,2] <- 5; x[2,3] <- 6;
> x

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

> matrix(1:6, nrow=2)

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

> matrix(1:6, nrow=2, byrow=FALSE)

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

> x[1,]

[1] 1 2 3

> x[,1]

[1] 1 4
```

R 매트릭스 행/열 추가

- ❏ `rbind` : 매트릭스 행 방향 합치기
- ❏ `cbind` : 매트릭스 열 방향 합치기
- ❏ 크기가 맞지 않으면 크기가 작은 인수를 재사용 (recycle)

```
> rbind(1:3, 4:6)

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

> cbind(1, 1:3)

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

> x <- matrix(1:6, nrow=2)
> x <- rbind(x, 7:9)
> x

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

행렬 연산

연산	의미	연산	의미
`t(A)`	전치 행렬 A'	`solve(A, b)`	x = Ab의 해
`A * B`	원소 단위 곱셈	`A %*% B`	행렬 곱셈
`A %o% B`	외적 AB'	`crossprod(A,B)`	내적 A'B
`diag(k)` (스칼라 k)	대각성분 k 단위행렬	`diag(x)` (벡터 x)	대각성분 x를 가지는 대각행렬
`diag(A)` (행렬 A)	행렬 A의 대각성분	`det(A)`	행렬식 (determinant)
`eigen(A)`	고유 (Eigenvalue) 분해	`svd(A)`	특이치 (singular value) 분해
`qr(A)`	QR분해 (QR Decomposition)	`chol(A)`	콜레스키 (Choleski) 분해

R 어레이

- `array(data = NA, dim = length(data), dimnames = NULL)` : 어레이 생성
- `x, dim, dimnames`는 각각 초기화 데이터, 차원벡터, 차원이름

```
> array(1:3, c(2,4))
     [,1] [,2] [,3] [,4]
[1,]    1    3    2    1
[2,]    2    1    3    2
```

차원이름 (dimnames)

- ❏ `rownames` : 매트릭스의 행 이름 설정 가능
- ❏ `colnames` : 매트릭스의 열 이름 설정 가능
- ❏ `dimnames` : 매트릭스, 어레이의 각 차원에 대한 이름 설정 가능 (리스트 자료형)

```
> x <- matrix(1:6, c(2,3))
> rownames(x) <- paste("row", 1:2)
> colnames(x) <- paste("col", 1:3)
> x

      col 1 col 2 col 3
row 1     1     3     5
row 2     2     4     6

> dimnames(x)

[[1]]
[1] "row 1" "row 2"

[[2]]
[1] "col 1" "col 2" "col 3"
```

R 리스트

- 벡터는 같은 자료형의 모임. 리스트는 다른 자료형의 모임
- `list()` 명령으로 생성. 보통 원소 이름을 넣지만 필수는 아니다.

```
> x <- list(name="Fred", wife="Mary", no.children=3, child.ages=c(4,7,9))
> x

$name
[1] "Fred"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9

> names(x)

[1] "name"         "wife"         "no.children" "child.ages"

> y <- list(1, "a")
> y

[[1]]
[1] 1

[[2]]
[1] "a"
```

R 리스트의 인덱싱

- ❏ 리스트의 단일 원소 인덱싱은 $ [[]] 기호를 이용한다.
- ❏ 리스트의 원소는 순서를 가지므로 이름이 아닌 정수로도 인덱싱 가능
- ❏ [] 기호로 인덱싱하면 원소가 아닌 부분집합인 리스트를 출력한다.

```
> x <- list(name="Fred", wife="Mary", no.children=3, child.ages=c(4,7,9))
> x$name
[1] "Fred"
> x$"name"
[1] "Fred"
> x[["name"]]
[1] "Fred"
> typeof(x[[1]])
[1] "character"
> x["name"]
$name
[1] "Fred"
> x[1]
$name
[1] "Fred"
> typeof(x[1])
[1] "list"
```

R 리스트 원소 추가/삭제

- 리스트의 원소 추가/삭제는 인덱싱을 이용
- 삭제시에는 해당 원소에 NULL 값을 설정

```
> x <- list(name="Fred", wife="Mary", no.children=3, child.ages=c(4,7,9))
> x$age <- 40
> x

$name
[1] "Fred"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9

$age
[1] 40

> x$wife <- NULL
> x

$name
[1] "Fred"

$no.children
[1] 3

$child.ages
[1] 4 7 9

$age
[1] 40
```

R 팩터

- 팩터는 유한개의 원소를 가지는 상태값(finite state)을 나타내는 자료형
- 실제로는 정수 벡터이지만 levels 속성값을 표시되는 문자열로 표현
- `levels()` 명령어 : level 이름을 표시/변경
- `nlevels()` 명령어 : level 수

```
> bloodtype <- factor(c("A", "B", "O", "A", "AB", "O", "A", "B"))
> bloodtype

[1] A  B  O  A  AB O  A  B
Levels: A AB B O

> attributes(bloodtype)

$levels
[1] "A"  "AB" "B"  "O"

$class
[1] "factor"

> unclass(bloodtype)

[1] 1 3 4 1 2 4 1 3
attr(,"levels")
[1] "A"  "AB" "B"  "O"

> levels(bloodtype)

[1] "A"  "AB" "B"  "O"

> levels(bloodtype) <- c("A type", "AB type", "B type", "O type")
> bloodtype

[1] A type  B type  O type  A type  AB type O type  A type
[8] B type
Levels: A type AB type B type O type

> nlevels(bloodtype)

[1] 4
```

R 데이터프레임

- 데이터 프레임은 길이가 같은 벡터들을 원소로 가지는 리스트
- R 통계 패키지에서 가장 일반적으로 사용되는 자료형
- `data.frame()` 명령어 사용. 문자열이 팩터로 변환되는 것을 막으려면 `stringsAsFactors=FALSE` 지정 혹은 인수에 `I()` 명령 사용

```
> x <- 1:3
> y <- c("A", "B", "C")
> z <- c(T,T,F)
> df <- data.frame(x,I(y),z,stringsAsFactors=FALSE)
> df

  x y     z
1 1 A  TRUE
2 2 B  TRUE
3 3 C FALSE

> names(df)

[1] "x" "y" "z"

> class(df)

[1] "data.frame"

> dim(df)

[1] 3 3
```

R 데이터프레임 인덱싱

- 데이터 프레임은 행방향으로는 벡터 특성, 열방향으로는 리스트 특성을 가짐.
  - `df$column1` 또는 `df[['column1']]` 인덱싱 가능, 결과는 벡터
- 매트릭스와 같은 `[row, col]` 인덱싱 사용 가능
  - `df[, 'column1']` : 열방향 슬라이싱, 결과는 벡터
  - `df['column1']` : 열방향 슬라이싱, 결과는 데이터프레임
  - `df[1, ]` : 행방행 슬라이싱, 결과는 데이터프레임

```
> df <- data.frame(matrix(1:6, nrow=2))
> df

  X1 X2 X3
1  1  3  5
2  2  4  6

> df$X1

[1] 1 2

> df[['X1']]

[1] 1 2

> df[,'X1']

[1] 1 2

> class(df[,'X1'])

[1] "integer"

> df['X1']

  X1
1  1
2  2

> class(df['X1'])

[1] "data.frame"
```

R 데이터프레임 검색

❑ 특정 조건을 만족하는 행(row)만 슬라이싱 가능

```
> df <- data.frame(matrix(1:20, ncol=2))
> df[(df$X1 > 4) & (df$X2 %% 2 == 0),]
   X1 X2
6   6 16
8   8 18
10 10 20
```

R 데이터프레임 행/열 증가

❑ 행/열 증가는 `rbind()`, `cbind()` 사용

```
> df <- data.frame(matrix(1:6, nrow=3))
> rbind(df, c(10,11))
   X1 X2
1   1  4
2   2  5
3   3  6
4  10 11
> cbind(df, X3=c(10,11,12))
   X1 X2 X3
1   1  4 10
2   2  5 11
3   3  6 12
```