1주차

R기초 & 데이터 구조

교육부 운영진

목 차

1. R 소개

•R은 오픈소스 프로그램으로 통계/데이터 마이닝 및 시각화를 위한 인터프리터 언어

•주로 연구 및 산업별 응용 프로그램으로 많이 사용되고 있으며 최근에는 기업들이 많이들 사용하기 시작함.

•Windows, Unix, macOS 등 다양한 운영체제 지원

•빅테이터 분석을 목적으로 하고 있으며 무료로 사용가능하며, 12000개가 넘는 패키지들이 다양한 기능을 지원하고 있으며 수시로 업데이트 되고 있음.

1. R 소개

Contents

Contents

Contents

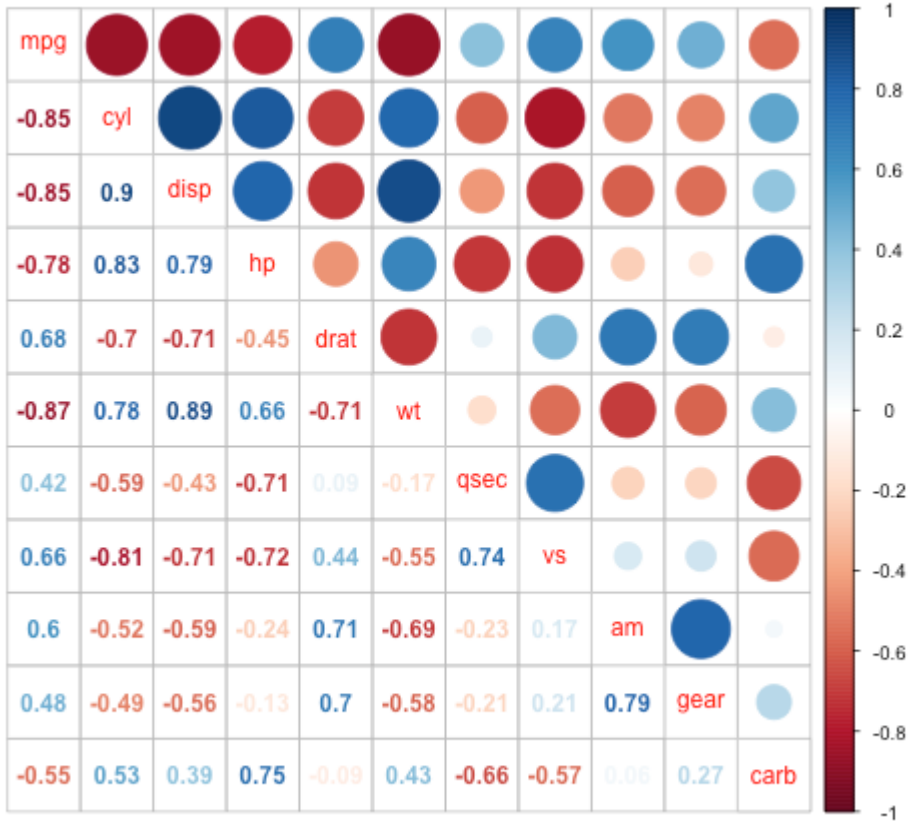
Contents

Contents

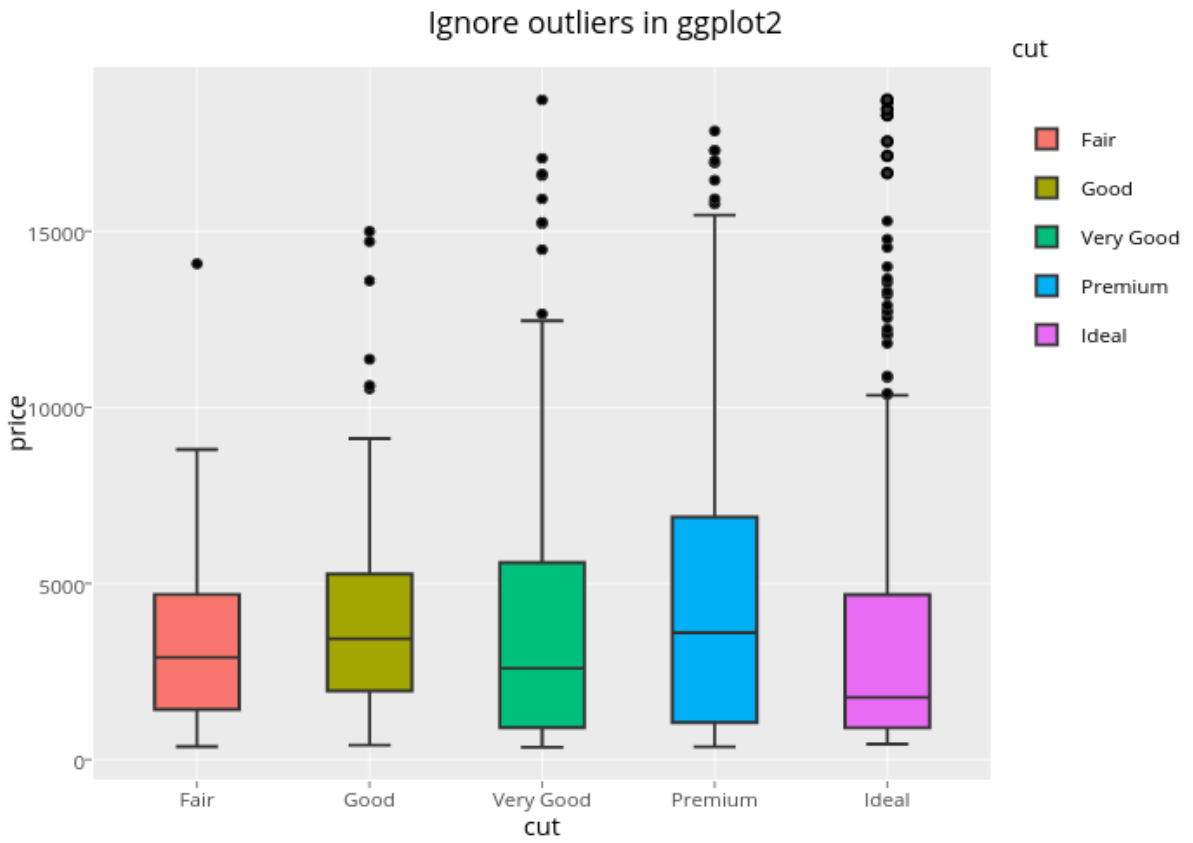
Contents

Contents

상관 행렬 (Correlation Matrix)



변수 시각화



1. R 소개

Contents

Contents

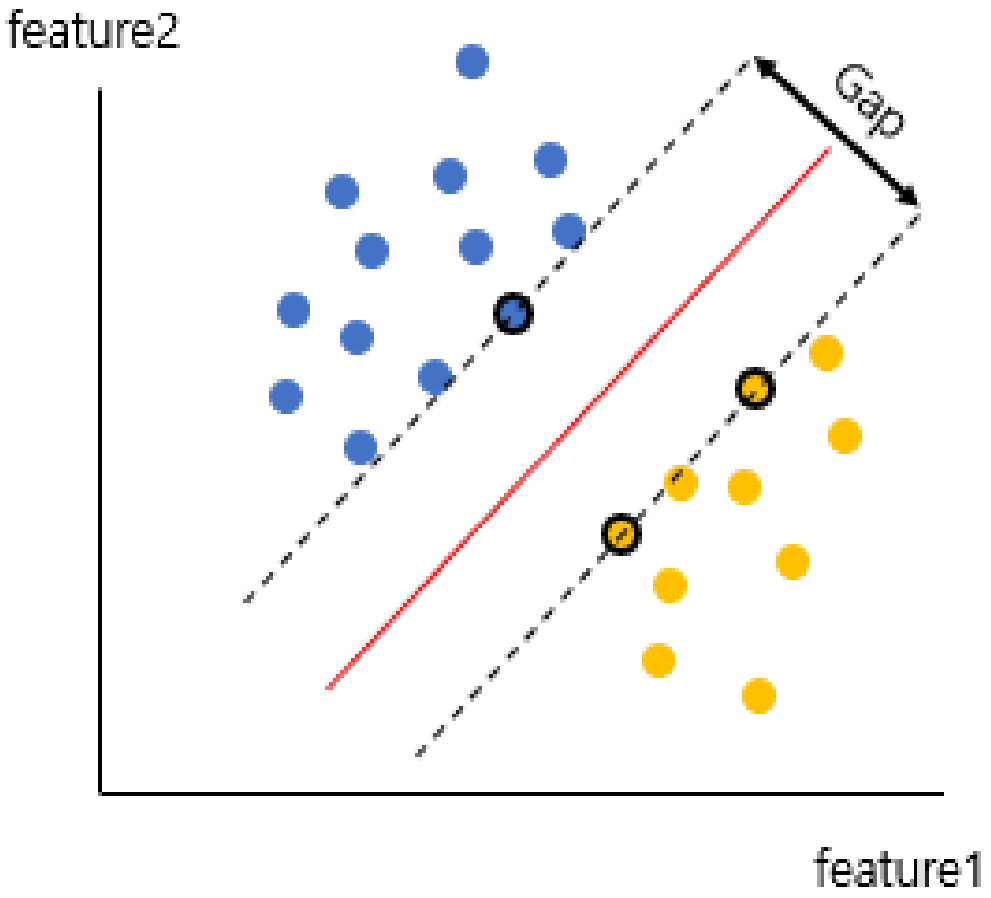
Contents

Contents

Contents

Contents

머신러닝



텍스트 마이닝

1. R 소개

Contents

Contents

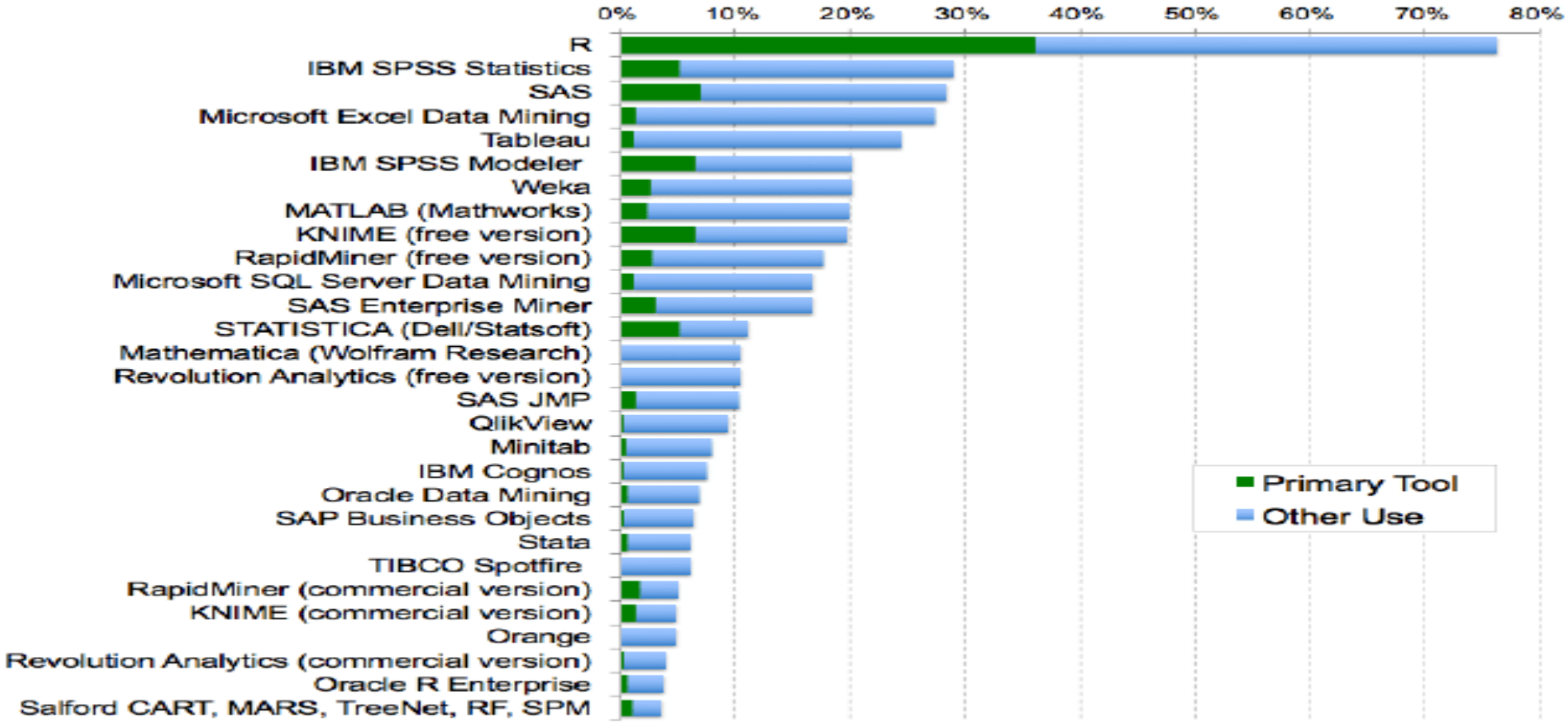
Contents

Contents

Contents

Contents

통계 프로그램 순위



1.1 R 기본 사용법

Contents

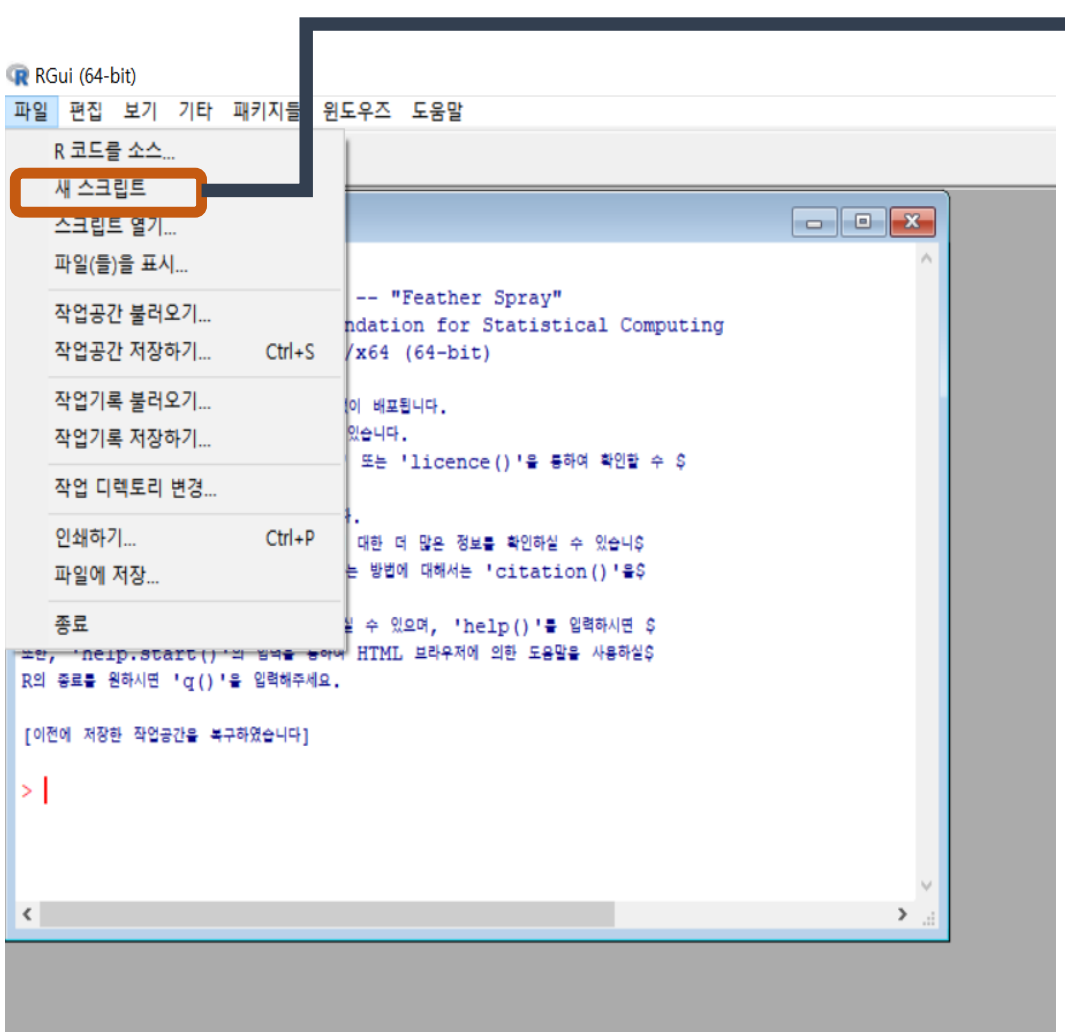
Contents

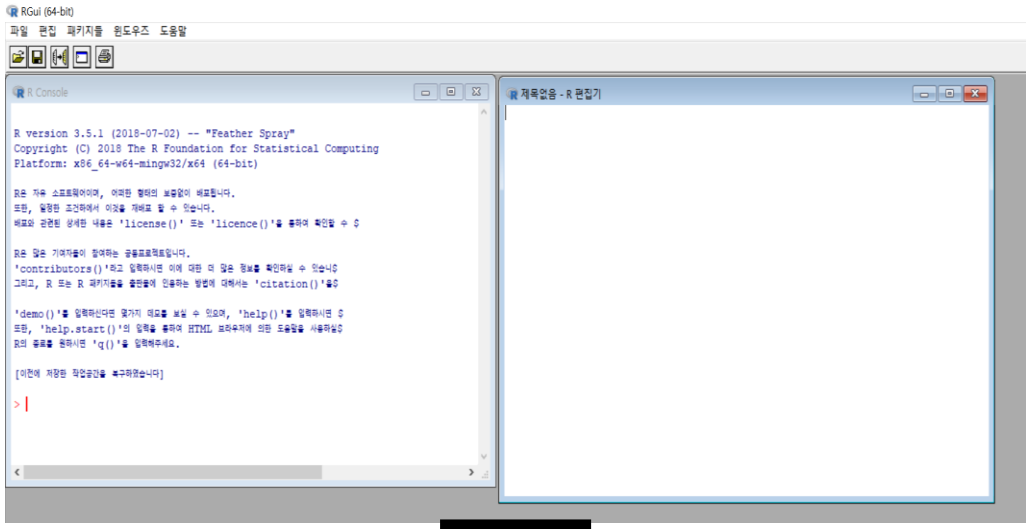
Contents

Contents

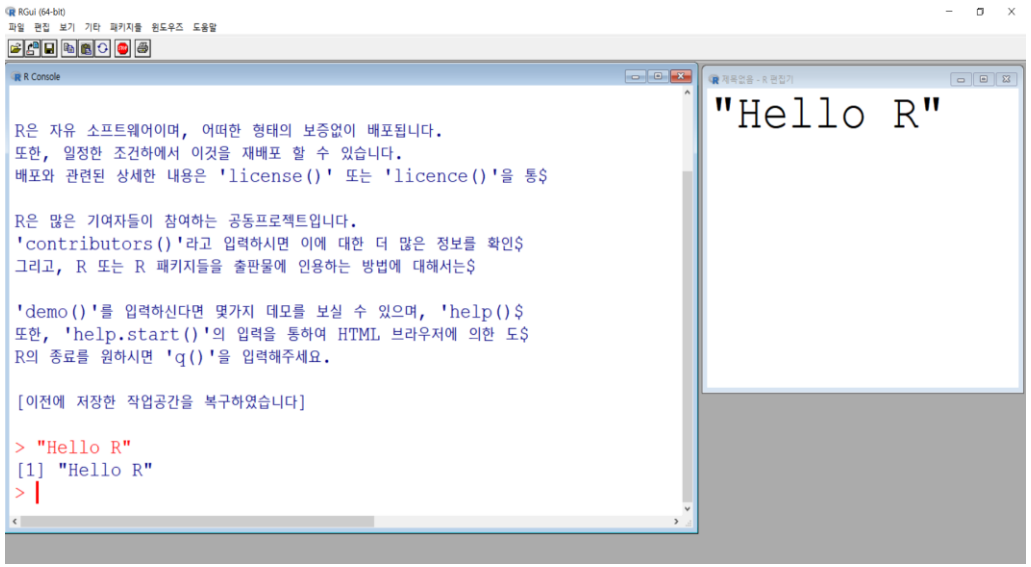
Contents

Contents

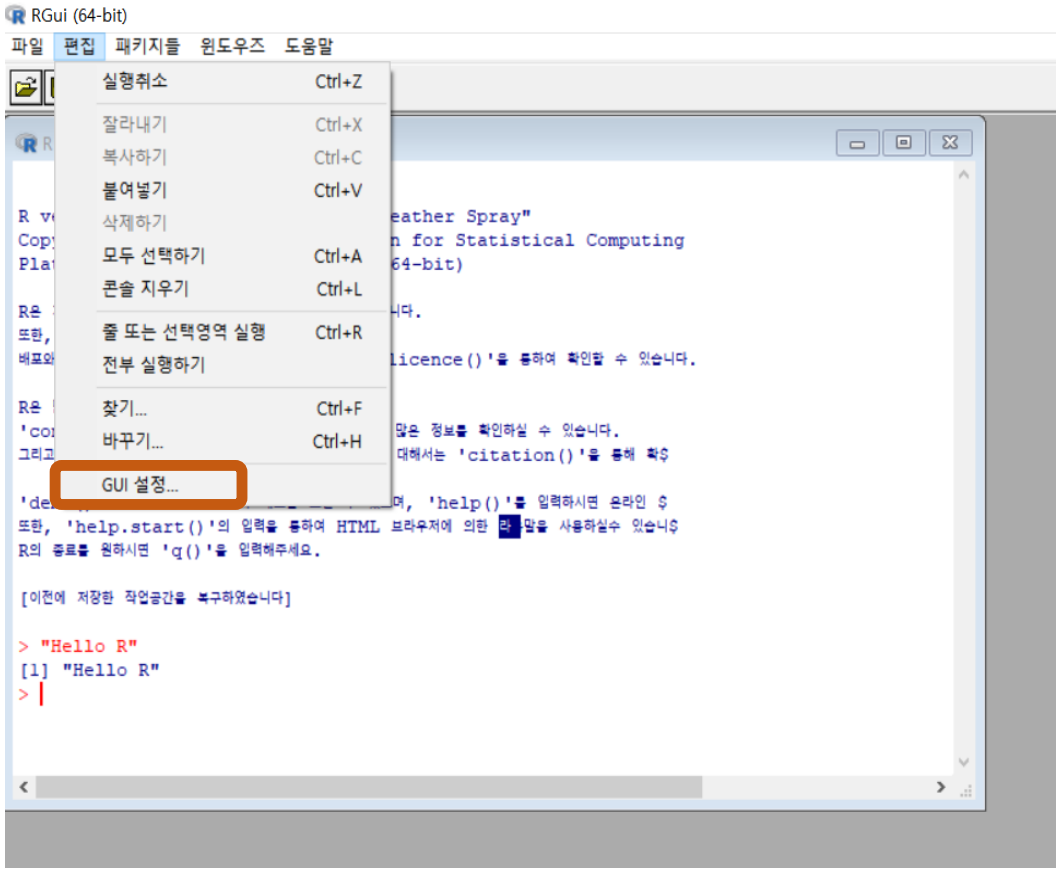




F5로 코드 실행



1.1 R 기본 사용법


RGui (64-bit)
파일 편집 패키지들 윈도우즈 도움말
실행취소 Ctrl+Z
잘라내기 Ctrl+X
복사하기 Ctrl+C
붙여넣기 Ctrl+V
삭제하기
모두 선택하기 Ctrl+A
콘솔 지우기 Ctrl+L
줄 또는 선택영역 실행 Ctrl+R
전부 실행하기
찾기... Ctrl+F
바꾸기... Ctrl+H
GUI 설정...
R version 3.5.1 (2018-07
Copyright (C) 2018 The R
Platform: x86_64-w64-min
'contributors()'라고 입력하시
'demo()'를 입력하신다면 몇가지 데
'help.start()'의 입력을
R의 종료를 원하시면 'q()'을 입력해
[이전에 저장한 작업공간을 복구하였습니다]
> "Hello R"
[1] "Hello R"
>
Rgui 구성 편집기
Single or multiple
MDI
SDI
MDI toolbar
MDI statusbar
Pager style
multiple windows
single window
Language for menus and messages
Font
Courier New
TrueType only
size
10
style
normal
Console rows
24
columns
85
Initial left
0
top
0
set options(width) on resize?
buffer console by default?
buffer chars
250000
lines
8000
Cursor blink
Partial
Pager rows
25
columns
80
Graphics windows: initial left
-25
top
0
Console and Pager Colours
background
normaltext
usertext
pagerbg
wheat2
wheat3
wheat4
white
Sample text
Apply
Save...
Load...
OK
Cancel

1.2 R STUDIO

Contents

Contents

Contents

Contents

Contents

Contents

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project: (None)

Untitled1* | airquality

Source on Save | Run | Source

```
1 plot(airquality$Ozone)
2
```

Ctrl+Enter로 실행

2:1 (Top Level) R Script

Console | Terminal

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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]

> View(airquality)
> View(airquality)
> plot(airquality$Ozone)
>
```

Environment | History | Connections

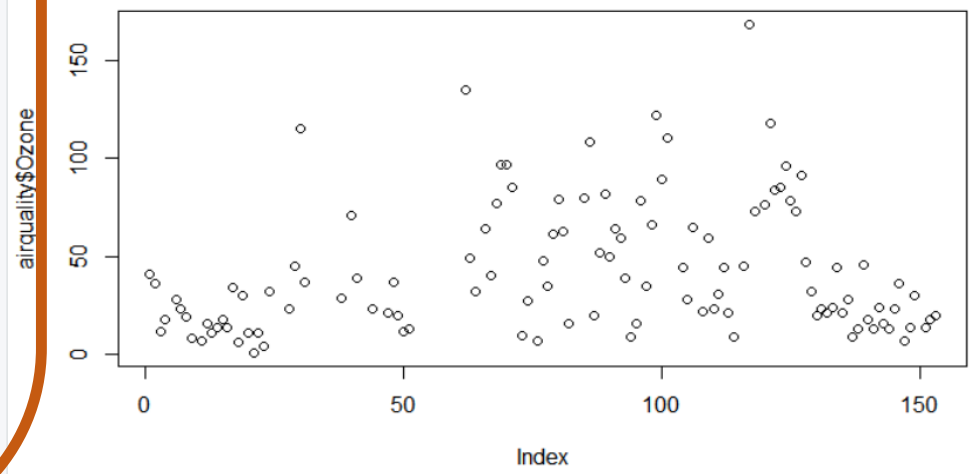
Import Dataset | List

Global Environment

Data

a	2956 obs. of 3 variables
irquality	153 obs. of 6 variables
runo	1150 obs. of 2 variables
ugu	num [1:9, 1:9] 1 2 3 4 5 6 7 8 9 2 ...
ris	150 obs. of 4 variables
onth6	0 obs. of 6 variables
onth7	0 obs. of 6 variables
onth8	0 obs. of 6 variables
ummer	0 obs. of 6 variables

Zoom | Export | Publish



2.1 학습하기 전에 알아두면 좋은 내용

Contents

Contents

Contents

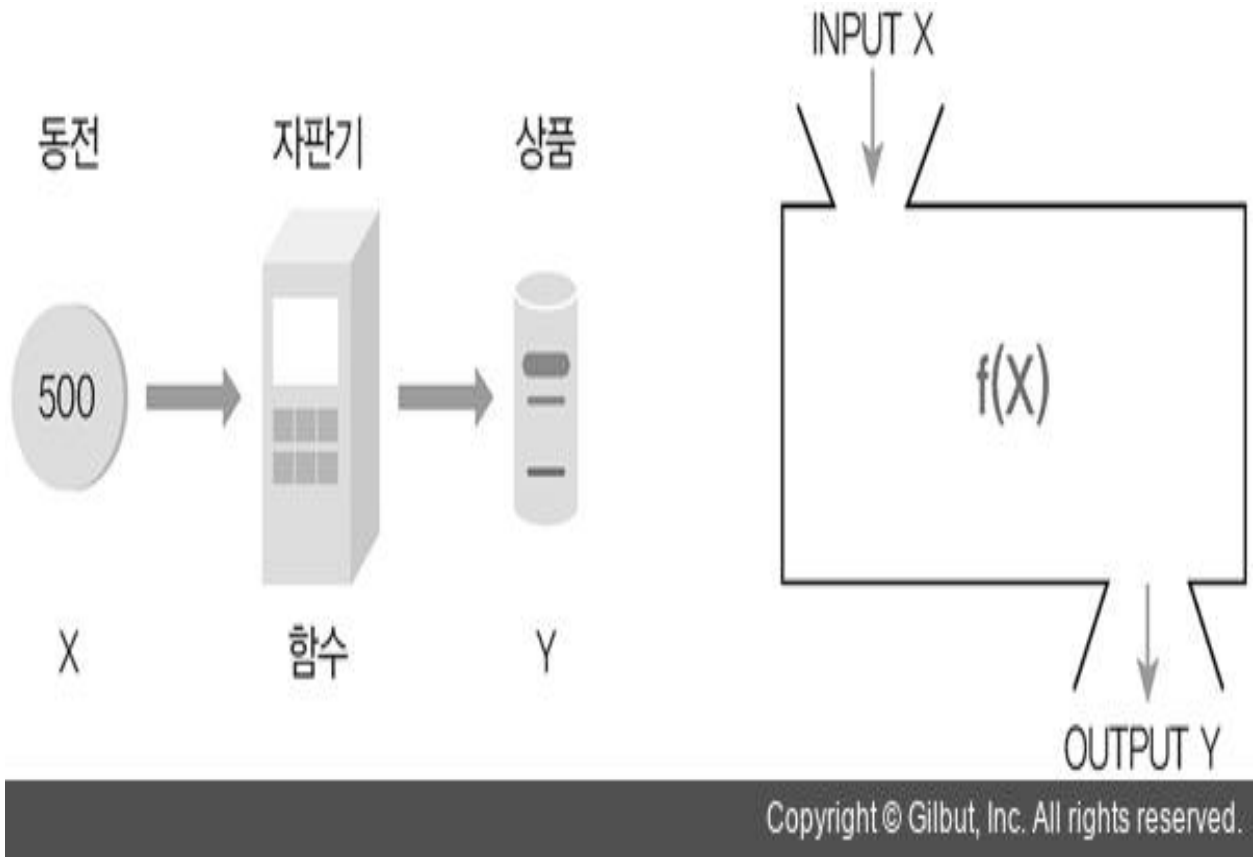
Contents

Contents

Contents

Contents

함수



수식,반복적인 작업 등 자주 사용되는 기능을 함수로 만들어 재사용

상품 <- 자판기(동전,지폐)

```
> round(10.2345,1)
[1] 10.2
```

2.1 학습하기 전에 알아두면 좋은 내용

Contents

Contents

Contents

Contents

Contents

Contents

주석 – '#'을통해 설명 및 주의사항등을 덧붙힘

```
>
> round(10.2345,1) 반올림 처리하는 함수
Error: unexpected symbol in "round(10.2345,1) 반올림"
> round(10.2345,1) #반올림 처리하는 함수
[1] 10.2
> #반올림 처리하는 함수
> round(10.2345,1)
[1] 10.2
```

#을 넣지 않아 명령어로 인식해 오류발생

대소문자 구분

```
> round(10.2345,1)
[1] 10.2
> ROUND(10.2345,1)
Error in ROUND(10.2345, 1) : could not find function "ROUND"
> Round(10.2345,1)
Error in Round(10.2345, 1) : could not find function "Round"
```

함수 이름은 unique 하며 대소문자 하나라도 틀려지면 함수가 실행되지 않음

2.1 학습하기 전에 알아두면 좋은 내용

Contents

Contents

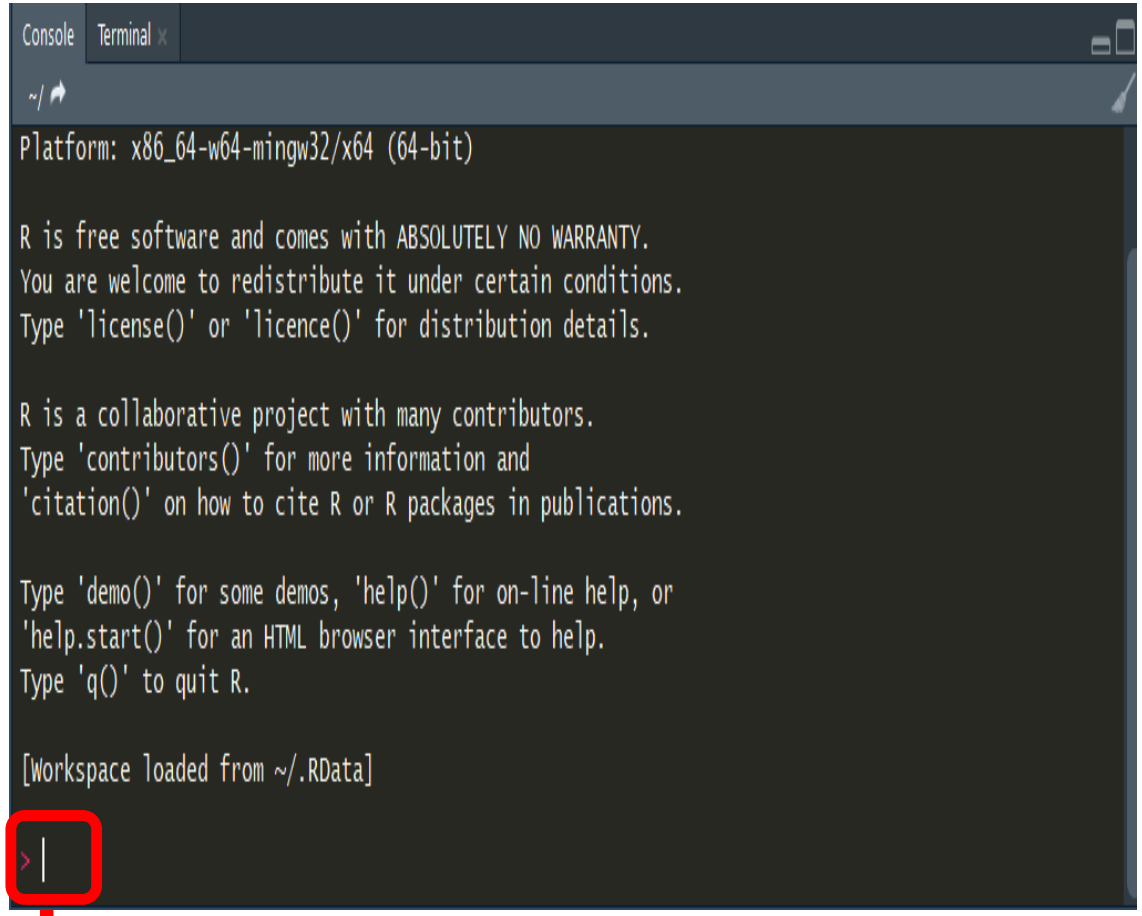
Contents

Contents

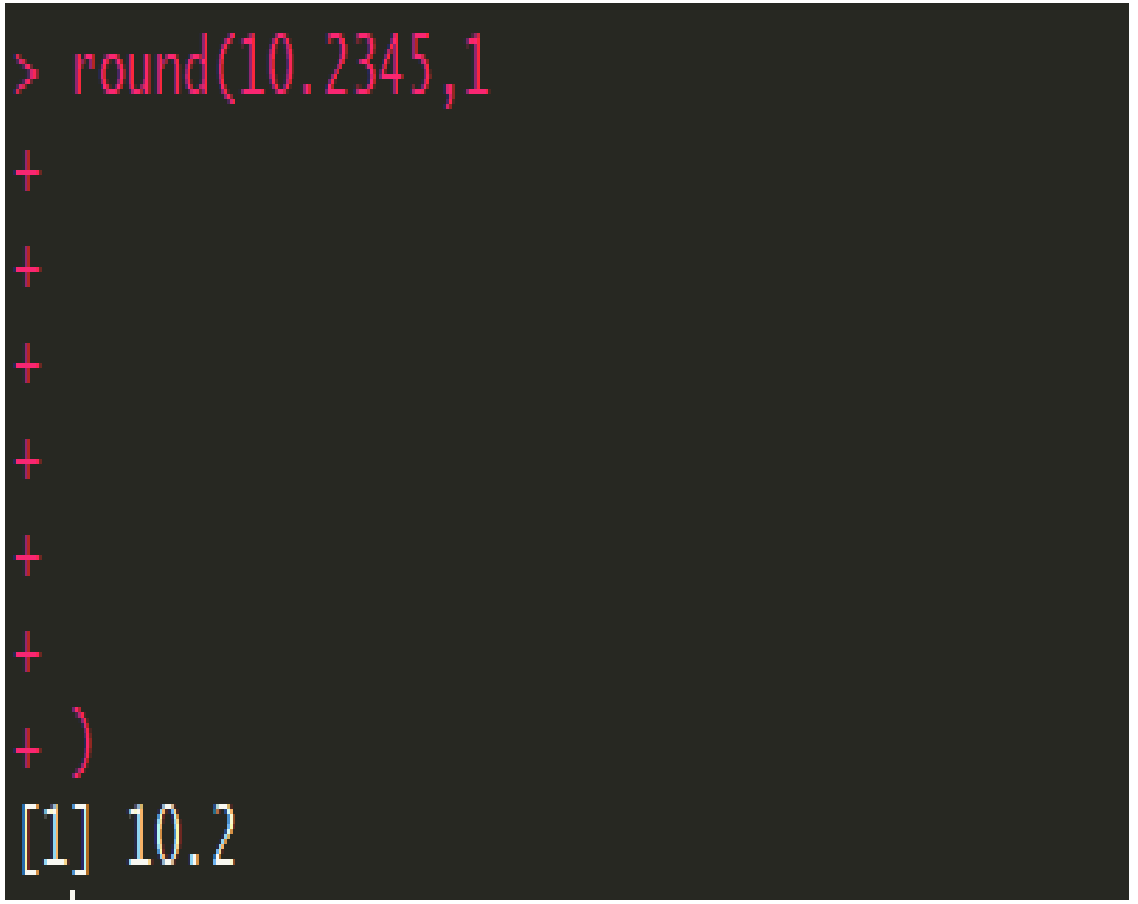
Contents

Contents

프롬포트



명령어를 수행할 준비가 됐음을 표시



명령어가 종료되지 않음 (보통 괄호나 따옴표가 안닫히는 경우) -> ')'를 입력하여 함수 실행
원인을 찾기 어려운 경우 ESC

2.2 너는 내가 정의한다

Contents

Contents

Contents

Contents

Contents

Contents

'<'와 '-'을 결합한 "<-" 기호

R 코드

```
16  #아무것도 설정돼 있지 않은 love 객체 출력
17  print(love)
18  #love 객체에 1을 지정
19  love<-1
20  #love값 출력
21  print(love)
22  #love 객체에 문자열 덮어쓰기
23  love<-"안녕하세요"
24  #love 값 출력(print를 이용하지않아도 출력가능)
25  love
```

-문자열의 경우 " "를 씌워줘야 함

Run(실행)

```
> #아무것도 설정돼 있지 않은 love 객체 출력
> print(love)
Error in print(love) : object 'love' not found
> #love 객체에 1을 지정
> love<-1
> #love값 출력
> print(love)
[1] 1
> #love 객체에 문자열 덮어쓰기
> love<-"안녕하세요"
> #love 값 출력(print를 이용하지않아도 출력가능)
> love
[1] "안녕하세요"
```

-지정된 값이 없어 에러 발생.

-print 함수는 입력으로 받은 객체를 출력하는 함수이나 생략가능

2.2 너는 내가 정의한다

Contents

'<'와 '-'을 결합한 "<-" 기호

R 코드

```
29  #love 변수에 print 함수를 지정
30  love<-print
31  #print 함수 대신 love로도 출력 가능
32  love("이제 나는 함수가 되었다!")
```

- 객체에 함수도 저장 가능

Run(실행)

```
> #love 변수에 print 함수를 지정
> love<-print
> #print 함수 대신 love로도 출력 가능
> love("이제 나는 함수가 되었다!")
[1] "이제 나는 함수가 되었다!"
```

- love 라는 객체가 print 와 같은 기능을 하는 함수가 됨.

2.2 너는 내가 정의한다

Contents

Contents

Contents

Contents

Contents

Contents

Contents

객체의 데이터 타입과 값의 일부를 출력하는 함수 str

R 코드

```
34  #객체 설정
35  love_num<-1
36  love_str<-"안녕하세요"
37  love_vec<-c(1,1,1,1)
38  love_fun<-print
39
40  #str 함수를 통해 객체의 정보를 확인
41  str(love_num)
42  str(love_str)
43  str(love_vec)
44  str(love_fun)
```

Run(실행)

```
> #str 함수를 통해 객체의 정보를 확인
> str(love_num)
 num 1
> str(love_str)
 chr "안녕하세요"
> str(love_vec)
 num [1:4] 1 1 1 1
> str(love_fun)
function (x, ...)
```

- 각 객체별 데이터 타입과 값이 나옴.
- num : 숫자형 변수
  chr : 문자형 변수
  function : 함수

2.2 너는 내가 정의한다

Contents

Contents

Contents

Contents

Contents

Contents

Contents

객체 유형

- **수치형(numeric)** : 숫자로 구성, 정수형과 실수형으로 구분
  ex) 3+5
- **논리형(logical)** : 참(TRUE)이나 거짓(FALSE)의 논리값 (반드시 대문자 사용)
  ex) 5<8
- **문자형(character)** : 문자나 문자열(“ “사용)
  ex) “Hi”
- **복소수형(complex)** : 실수와 허수로 구성된 복소수
  ex) 1+4i

2.2 너는 내가 정의한다

Contents

Contents

Contents

Contents

Contents

Contents

Contents

특수 객체 유형

- **NULL** : 비어있는 값
- **NA** : 결측치(Not Available/Missing value)
- **NaN** : 수학적으로 정의가 불가능한 수(Not a Number)
  ex) sqrt(-3), log(-5), 0/0
- **Inf** : 양의 무한대(Infinity number)
  ex) 1/0
- **-Inf** : 음의 무한대

- **NULL은 비어있는 객체를 말함**
- **NA는 결측치, 즉 데이터가 존재 하지 않음을 의미**

2.2 너는 내가 정의한다

Contents

Contents

Contents

Contents

Contents

Contents

Contents

이름이 다른 객체는 독립된 객체

R 코드

```
42  # B는 A로부터 지정된 객체
43  A<-1
44  B<-A
45  A ; B
46
47  #A값이 변경되어도 B는 여전히 1
48  A=9
49  A ; B
```

Run(실행)

```
> # B는 A로부터 지정된 객체
> A<-1
> B<-A
> A ; B
[1] 1
[1] 1
>
> #A값이 변경되어도 B는 여전히 1
> A=9
> A ; B
[1] 9
[1] 1
```

- A ; B : ";"를 통해 A와 B를 동시에 실행
- "<-" 대신 "=" 로도 객체 지정 가능

- A, B 순서대로 출력

- B는 변하지 않음

2.2 너는 내가 정의한다

Contents

Contents

Contents

Contents

Contents

Contents

객체 지정시 주의사항

- 일반적으로 문자, 숫자, “_”, “.”의 조합으로 지을 수 있음

- 첫 글자는 문자나 “.” 으로 시작할 수 있음.

- 첫글자는 “_” 및 숫자 X

- 단, “.”으로 시작할 경우 두 번째 문자에는 숫자가 올 수 없음.

- 특정 키워드는 사용 X

  ex) if <- c(1,2,3)

객체 지정 예시) iris_tmp, mean_tmp, obj_3 등등

2.3 데이터 구조의 기본-벡터

Contents

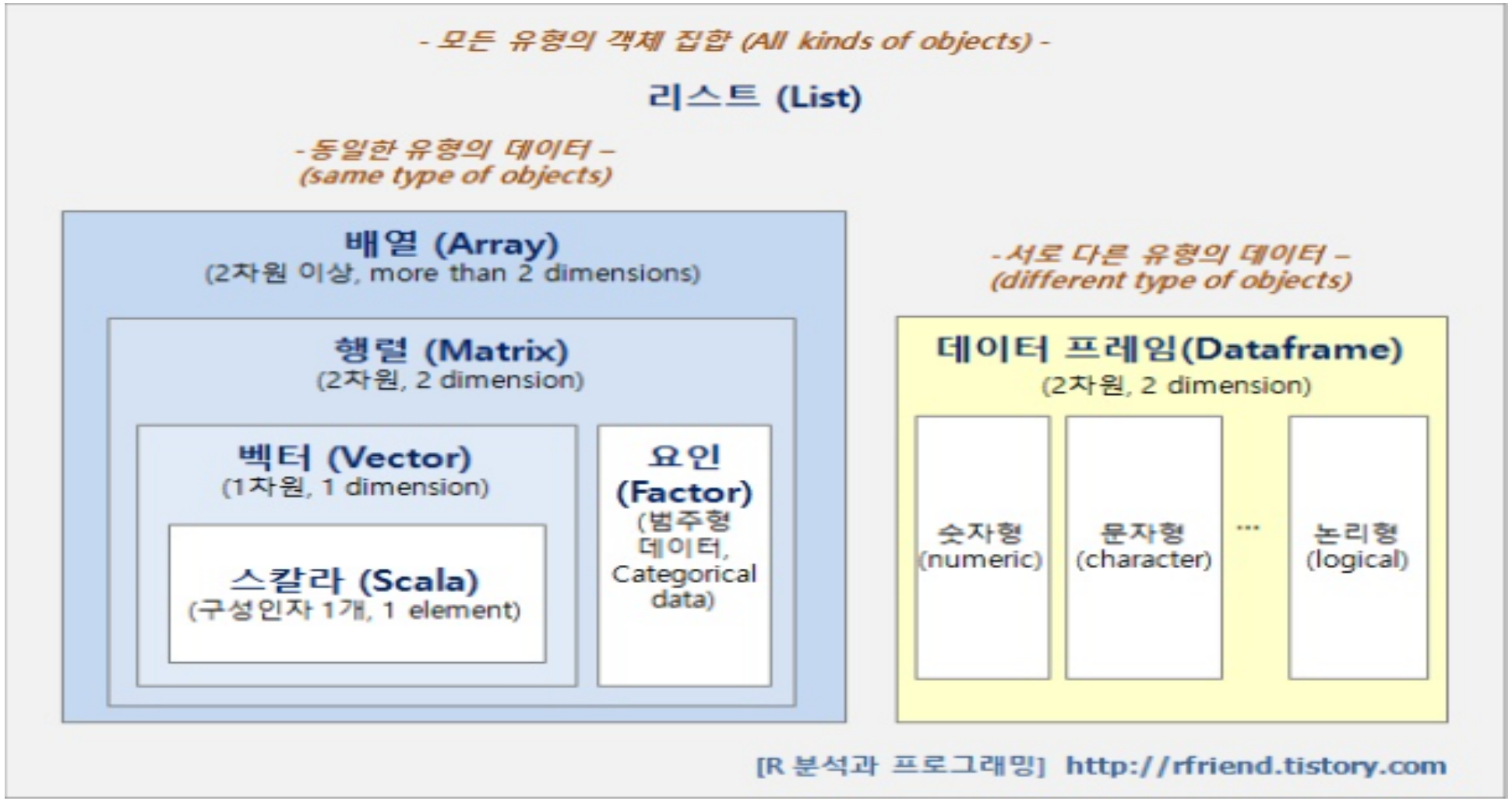
Contents

Contents

Contents

Contents

Contents



2.3.1 벡터 생성

Contents

Contents

Contents

Contents

Contents

Contents

벡터 생성

R 코드

```
51  # 1, 2, 3, 4를 요소로 가지는 벡터 생성
52  vec_t<-c(1,2,3,4)
53  #생성된 벡터의 값 확인
54  vec_t
55  #벡터 정보 확인 - 길이가 4인 숫자 벡터 / 값은 1, 2, 3, 4
56  str(vec_t)
57  #벡터의 길이 확인
58  length(vec_t)
```

- “c”를 이용하여 벡터 생성
- length() : 길이확인

Run(실행)

```
> # 1, 2, 3, 4를 요소로 가지는 벡터 생성
> vec_t<-c(1,2,3,4)
> #생성된 벡터의 값 확인
> vec_t
[1] 1 2 3 4
> #벡터 정보 확인 - 길이가 4인 숫자 벡터 / 값은 1, 2, 3, 4
> str(vec_t)
 num [1:4] 1 2 3 4
> #벡터의 길이 확인
> length(vec_t)
[1] 4
```

- Str 함수를 이용하여 num 유형임을 파악
- length 함수로 벡터의 길이 4가 출력 됨

2.3.1 벡터 생성

Contents

Contents

Contents

Contents

Contents

Contents

여러가지 데이터 타입을 벡터로 생성할 경우

R 코드

```r
#벡터를 생성할 때 문자와 숫자를 함께 사용하면?
vec_t<-c(1,"hi",2)
vec_t
#벡터 정보 확인
str(vec_t)
```

Run(실행)

```
> #벡터를 생성할 때 문자와 숫자를 함께 사용하면?
> vec_t<-c(1,"hi",2)
> vec_t
[1] "1"  "hi" "2"
> #벡터 정보 확인
> str(vec_t)
 chr [1:3] "1" "hi" "2"
```

- 벡터는 모두 같은 데이터 타입의 요소를 추가하여야 함.

- 숫자와 문자를 함께 담으면 숫자를 문자로 변환하여 문자 벡터로 생성

- 1과 2에 “ “ 가 씌여져 있음을 확인할 수 있음

2.3.2 벡터의 데이터 타입

Contents

2.3.2.1 숫자형 벡터

R 코드

```
67  # 숫자형 벡터
68  numeric_vector<-c(0.2,-1,2,-0.5)
69  # numeric_vector의 타입 확인
70  mode(numeric_vector)
71  # 숫자 벡터를 연산하는 다양한 함수
72  min(numeric_vector) ; max(numeric_vector) ; mean(numeric_vector) ;median(numeric_vector)
73  sum(numeric_vector)
```

Run(실행)

```
> # 숫자형 벡터
> numeric_vector<-c(0.2,-1,2,-0.5)
> # numeric_vector의 타입 확인
> mode(numeric_vector)
[1] "numeric"
> # 숫자 벡터를 연산하는 다양한 함수
> min(numeric_vector) ; max(numeric_vector) ; mean(numeric_vector) ;median(numeric_vector)
[1] -1
[1] 2
[1] 0.175
[1] -0.15
> sum(numeric_vector)
[1] 0.7
```

- **mode() : 벡터의 타입을 확인하는 함수**
- **min() : 최솟값 max() : 최댓값**
  mean() : 평균 median() : 중위수
  sum() : 합계

- **벡터의 값들에 대해서 각 함수를 적용하여 연산 가능.**

2.3.2 벡터의 데이터 타입

Contents

Contents

Contents

Contents

Contents

Contents

Contents

2.3.2.1 숫자형 벡터 특수 유형

R 코드

```
75  # Inf 와 NaN이 포함된 벡터 만들기
76  numeric_vector2<-c(1/0,2/2,-2/2,-1/0,0/0)
77  numeric_vector2
```

Run(실행)

```
> # Inf 와 NaN이 포함된 벡터 만들기
> numeric_vector2<-c(1/0,2/2,-2/2,-1/0,0/0)
> numeric_vector2
[1]  Inf    1   -1 -Inf  NaN
```

- **1/0 , -1/0 은 무한대로 나타나지만 0/0은 수학적으로 정의 할 수 없으므로 “Not a Number” 즉, NaN 으로 나타난다.**

2.3.2 벡터의 데이터 타입

Contents

Contents

Contents

Contents

Contents

Contents

Contents

2.3.2.2 논리형 벡터

R 코드

```
79  # TRUE, FALSE로 설정
80  ex_logical1<-c(TRUE,FALSE,TRUE,FALSE)
81  ex_logical1
82  mode(ex_logical1)
83  # T,F로 설정
84  ex_logical2<-c(T,F,T,F)
85  ex_logical2
86  mode(ex_logical2)
```

Run(실행)

```
> # TRUE, FALSE로 설정
> ex_logical1<-c(TRUE,FALSE,TRUE,FALSE)
> ex_logical1
[1]  TRUE FALSE  TRUE FALSE
> mode(ex_logical1)
[1] "logical"
> # T,F로 설정
> ex_logical2<-c(T,F,T,F)
> ex_logical2
[1]  TRUE FALSE  TRUE FALSE
> mode(ex_logical2)
[1] "logical"
```

- 논리형은 logical
- T,F 와 TRUE,FALSE는 같다.

2.3.2 벡터의 데이터 타입

Contents

Contents

Contents

Contents

Contents

Contents

Contents

2.3.2.2 논리형 벡터 오류

R 코드

```
88  # 소문자 오류
89  ex_logical3<-c(true,false,true,false) ; ex_logical3
90  # 따옴표를 붙이면 문자열로 인식
91  ex_logical4<-c("TRUE","FALSE","TRUE","FALSE") ;ex_logical4
92  mode(ex_logical4)
```

Run(실행)

```
> # 소문자 오류
> ex_logical3<-c(true,false,true,false) ; ex_logical3
Error: object 'true' not found
> # 따옴표를 붙이면 문자열로 인식
> ex_logical4<-c("TRUE","FALSE","TRUE","FALSE") ;ex_logical4
[1] "TRUE"  "FALSE" "TRUE"  "FALSE"
> mode(ex_logical4)
[1] "character"
```

- **소문자로 쓸 경우 객체로 인지함**
- **따옴표를 붙이면 문자열로 인식**

2.3.2 벡터의 데이터 타입

Contents

Contents

Contents

Contents

Contents

Contents

2.3.2.2 논리형 벡터 역변환

R 코드

```
94  ex_logical<-c(TRUE,T,FALSE,F)
95  ex_logical
96  # ! 기호를 이용해 역변환
97  !ex_logical
```

Run(실행)

```
> ex_logical<-c(TRUE,T,FALSE,F)
> ex_logical
[1]  TRUE  TRUE FALSE FALSE
> # ! 기호를 이용해 역변환
> !ex_logical
[1] FALSE FALSE  TRUE  TRUE
```

- ! 기호로 TRUE 는 FALSE로 FALSE 는 TRUE 로 바뀜
- != 기호를 이용하면 같지 않다 라는 의미로 사용 가능

2.3.2 벡터의 데이터 타입

Contents

Contents

Contents

Contents

Contents

Contents

Contents

2.3.2.2 논리 값과 숫자 값의 관계

R 코드

```
 99  # 0인 값만 FALSE로 인식
100  ex_logical<-as.logical(c(0,-1,1,100,-7)) ; ex_logical
101  # FALSE는 0, TRUE는 1로 변환
102  as.numeric(ex_logical)
```

- as.logical() : logical 변수로 변환
- as.numeric() : 숫자형으로 변환

Run(실행)

```
> # 0인 값만 FALSE로 인식
> ex_logical<-as.logical(c(0,-1,1,100,-7)) ; ex_logical
[1] FALSE  TRUE  TRUE  TRUE  TRUE
> # FALSE는 0, TRUE는 1로 변환
> as.numeric(ex_logical)
[1] 0 1 1 1 1
```

- R 내부적으로 0은 FALSE, 그 밖의 숫자는 TRUE 이다.
- 반대로 논리 값을 숫자 값으로 변환할 때는 FALSE는 0 TRUE는 1로 변환

2.3.2 벡터의 데이터 타입

Contents

Contents

Contents

Contents

Contents

Contents

2.3.2.3 문자열 벡터

R 코드

```r
# 문자열 벡터 생성
v_charater<-c("문자열","문자열2","A","1")
v_charater
mode(v_charater)
```

- "" 또는 ''를 활용해 문자열 표현

Run(실행)

```
> # 문자열 벡터 생성
> v_charater<-c("문자열","문자열2","A","1")
> v_charater
[1] "문자열"  "문자열2" "A"       "1"
> mode(v_charater)
[1] "character"
```

2.3.2 벡터의 데이터 타입

Contents

2.3.2.3 문자열 다루기

R 코드

```
109 # 문자 개수 출력
110 nchar(c("F123","F124","F125","F1227"))
111 #문자열 자르기 - 두 번째부터 네 번째 문자 사이의 문자열 추출
112 substr("1234567",2,4)
113 substr(c("F123","F124","F125","F1227"),2,4)
114 # 특정 문자로 데이터 나누기 - split에 정의한 구분자를 기준으로 문자열을 나누어 벡터로 변환
115 a<-strsplit('2014/11/22', split="/") ; a
116 a[[1]][1]
```

Run(실행)

```
> # 문자 개수 출력
> nchar(c("F123","F124","F125","F1227"))
[1] 4 4 4 5
> #문자열 자르기 - 두 번째부터 네 번째 문자 사이의 문자열 추출
> substr("1234567",2,4)
[1] "234"
> substr(c("F123","F124","F125","F1227"),2,4)
[1] "123" "124" "125" "122"
> # 특정 문자로 데이터 나누기 - split에 정의한 구분자를 기준으로 문자열을 나누어 벡터로 변환
> a<-strsplit('2014/11/22', split="/") ; a
[[1]]
[1] "2014" "11"   "22"

> a[[1]][1]
[1] "2014"
```

- nchar(x) : 문자열의 길이 출력
- substr(x,start,stop) : x를 시작지점부터 끝지점 까지 추출
- strsplit(x,split) : x를 split 구분자로 나누어 반환

- 이 함수들은 기본적인 분석은 물론, 텍스트 마이닝에도 많이 사용 되는 함수이다.

2.3.2 벡터의 데이터 타입

Contents

2.3.2.3 문자열 다루기

R 코드

```
119 # 문자열 합치기 - 합칠 때 문자열 사이의 문자는 sep에 정의
120 #                 sep을 정의하지 않으면 문자열 사이에 공백을 붙여 합침
121 paste("50=","30+","20")
122 paste("50=","30+","20",sep="")
123 paste("50","30","20",sep="*")
124 # 대문자 변환
125 toupper("AbCdEfGhIjKlMn")
126 # 소문자 변환
127 tolower("AbCdEfGhIjKlMn")
```

Run(실행)

```
> # 문자열 합치기 - 합칠 때 문자열 사이의 문자는 sep에 정의
> #                 sep을 정의하지 않으면 문자열 사이에 공백을 붙여 합침
> paste("50=","30+","20")
[1] "50= 30+ 20"
> paste("50=","30+","20",sep="")
[1] "50=30+20"
> paste("50","30","20",sep="*")
[1] "50*30*20"
> # 대문자 변환
> toupper("AbCdEfGhIjKlMn")
[1] "ABCDEFGHIJKLMN"
> # 소문자 변환
> tolower("AbCdEfGhIjKlMn")
[1] "abcdefghijklmn"
```

- **paste(... , sep) : ...에 문자열을 sep에 값으로 합친다.**
- **toupper(x) : x라는 문자열을 대문자로 변환**
- **tolower(x) : x라는 문자열을 소문자로 변환**

2.3.2 벡터의 데이터 타입

Contents

Contents

Contents

Contents

Contents

Contents

2.3.2.4 팩터(factor)

학번	과목	성적
S1234	영어	A
S1234	국어	C
S1234	사회	B
S5678	영어	C
S5678	국어	B
S5678	사회	A

- R에서는 범주형 데이터(Categorical Data)를 팩터(factor)로 표현
- 범주형 데이터 : 몸무게, 거리 등 연속된 데이터와 달리 제한된 범주(category) 안에서 선택되는 데이터
- ex) 성별, 수강과목, 혈액형 등

- 범주형 데이터는 “서열형 데이터”와 “명목형 데이터”로 나뉜다. 서열이 존재할 경우 “서열형 데이터”, 단순 분류의 경우 “명목형 데이터 " 로 구분한다.
- ex) 과목: 명목형 데이터, 성적: 서열형 데이터

2.3.2 벡터의 데이터 타입

Contents

2.3.2.4 factor

R 코드

```
130 # 팩터로 변환할 문자 벡터
131 v_character<-c("사과","복숭아","사과","오렌지","사과","오렌지","복숭아")
132 # factor 함수로 팩터를 생성
133 v_factor<-factor(v_character) ; v_factor
134 # 데이터 타입 확인 - 문자로 출력되나 데이터 타입은 숫자(numeric)
135 mode(v_factor)
136 #객체 정보 확인
137 str(v_factor)
```

```
> # 팩터를 문자 벡터로 변환
> v_factor_to_char<-as.character(v_factor) ; v_factor_to_char
[1] "사과"   "복숭아" "사과"   "오렌지" "사과"   "오렌지" "복숭아"
> # 팩터를 숫자 벡터로 변환
> v_factor_to_num<-as.numeric(v_factor) ; v_factor_to_num
[1] 2 1 2 3 2 3 1
```

Run(실행)

```
> # 팩터로 변환할 문자 벡터
> v_character<-c("사과","복숭아","사과","오렌지","사과","오렌지","복숭아")
> # factor 함수로 팩터를 생성
> v_factor<-factor(v_character) ; v_factor
[1] 사과   복숭아 사과   오렌지 사과   오렌지 복숭아
Levels: 복숭아 사과 오렌지
> # 데이터 타입 확인 - 문자로 출력되나 데이터 타입은 숫자(numeric)
> mode(v_factor)
[1] "numeric"
> #객체 정보 확인
> str(v_factor)
 Factor w/ 3 levels "복숭아","사과",..: 2 1 2 3 2 3 1
```

- factor(x, levels, ordered) : x를 팩터로 생성
- levels: x의 범주를 정의한 벡터
- ordered: 서열형 데이터인 경우 TURE로 설정 (default는 false /

내부적으로는 숫자로 된 코드 값 백터로 구성

- 이 때문에 같은 factor 객체라도 “문자 벡터”로 변환한 결과와 “숫자 벡터”로 변환한 결과가 다름

2.3.2 벡터의 데이터 타입

Contents

2.3.2.4 factor 올바른 변환

R 코드

```
162  # 숫자 벡터 올바른 변환 방법
163  v_num<-c(1000,2000,1000,2000,3000,2000,3000)
164  # 숫자 벡터를 팩터로 변환
165  v_num_factor<-factor(v_num) ; v_num_factor
166  # 팩터->문자 벡터로 변환
167  v_char<-as.character(v_num_factor) ; v_char
168  # 문자 벡터-> 숫자 벡터로 변환
169  v_num<-as.numeric(v_char) ; v_num
```

Run(실행)

```
> # 숫자 벡터
> v_num<-c(1000,2000,1000,2000,3000,2000,3000)
> # 숫자 벡터를 팩터로 변환
> v_num_factor<-factor(v_num) ; v_num_factor
[1] 1000 2000 1000 2000 3000 2000 3000
Levels: 1000 2000 3000
> # 팩터->문자 벡터로 변환
> v_char<-as.character(v_num_factor) ; v_char
[1] "1000" "2000" "1000" "2000" "3000" "2000" "3000"
> # 문자 벡터-> 숫자 벡터로 변환
> v_num<-as.numeric(v_char) ; v_num
[1] 1000 2000 1000 2000 3000 2000 3000
```

- 팩터에서 숫자 벡터로 바로 변환할 시 다른 값이 나오므로 문자 벡터로 먼저 변환한 후 숫자 벡터로 변환해야 한다

- 데이터 타입 변경 함수
- as.데이터형

  ex) as.numeric,as.character, as.data.frame, as.logical

2.3.2 벡터의 데이터 타입

Contents

2.3.2.3 factor levels 조정

R 코드

```
146 # 팩터로 변환할 문자 벡터
147 v_character<-c("사과","복숭아","사과","오렌지","사과","오렌지","복숭아")
148 # factor 함수로 팩터 객체 생성
149 v_factor<-factor(v_character,levels=c("사과","복숭아")) ; v_factor
150 # 범주 순서 변경
151 v_factor<-factor(v_character,levels=c("복숭아","오렌지","사과")) ; v_factor
```

Run(실행)

```
> # 팩터로 변환할 문자 벡터
> v_character<-c("사과","복숭아","사과","오렌지","사과","오렌지","복숭아")
> # factor 함수로 팩터 객체 생성
> v_factor<-factor(v_character,levels=c("사과","복숭아")) ; v_factor
[1] 사과   복숭아 사과   <NA>   사과   <NA>   복숭아
Levels: 사과 복숭아
> # 범주 순서 변경
> v_factor<-factor(v_character,levels=c("복숭아","오렌지","사과")) ; v_factor
[1] 사과   복숭아 사과   오렌지 사과   오렌지 복숭아
Levels: 복숭아 오렌지 사과
```

- Levels 를 통해 범주 및 순서 정의

"오렌지"는 정의 하지 않았으므로 NA로 표기됨

범주 순서는 그래프나 분석 결과가 출력될 때 영향을 줌

2.3.2 벡터의 데이터 타입

Contents

2.3.2.3 factor ordered 옵션

R 코드

```
154 # 등급을 나타내는 문자 벡터
155 ex_label<-c("하하","중하","중","중상","상상")
156 # 서열형 데이터 팩터 생성
157 ordered_factor<-factor(ex_label,ordered=T) ; ordered_factor
158 # levels 입력 항목을 이용해 서열 순으로 범주 순서를 정의
159 factor(ex_label, levels=c("하하","중하","중","중상","상상"),ordered=T)
```

Run(실행)

```
> # 등급을 나타내는 문자 벡터
> ex_label<-c("하하","중하","중","중상","상상")
> # 서열형 데이터 팩터 생성
> ordered_factor<-factor(ex_label,ordered=T) ; ordered_facto
[1] 하하 중하 중   중상 상상
Levels: 상상 < 중 < 중상 < 중하 < 하하
> # levels 입력 항목을 이용해 서열 순으로 범주 순서를 정의
> factor(ex_label, levels=c("하하","중하","중","중상","상상"),ordered=T)
[1] 하하 중하 중   중상 상상
Levels: 하하 < 중하 < 중 < 중상 < 상상
```

- ordered=T 를 통해 서열형 팩터 생성
- "<"기호를 통해 서열을 표시하나 범주 간의 서열을 별도로 정의하지 않아 서열이 잘못 표현
- levels를 통해 조절

2.3.2 벡터의 데이터 타입

Contents

table 함수 소개

R 코드

```
171  # 도수 분포표
172  table(mtcars$cyl)
173  # 두 개의 변수 설정 (교차표)
174  table(mtcars$cyl, mtcars$gear)
175  # 행, 열에 변수 이름 설ㅈ
176  table(mtcars$cyl, mtcars$gear, dnn=c("Cylinder", "Gear"))
```

Run(실행)

```
> # 도수 분포표
> table(mtcars$cyl)

 4  6  8
11  7 14
> # 두 개의 변수 설정 (교차표)
> table(mtcars$cyl, mtcars$gear)

     3  4  5
  4  1  8  2
  6  2  4  1
  8 12  0  2
> # 행, 열에 변수 이름 설ㅈ
> table(mtcars$cyl, mtcars$gear, dnn=c("Cylinder", "Gear"))
        Gear
Cylinder  3  4  5
       4  1  8  2
       6  2  4  1
       8 12  0  2
```

- **table(**... , dnn)
- **... : 한 개 이상의 객체를 설정, 한 개의 객체가 주어지면 도수 분포표, 두 개 이상의 객체가 주어지면 교차표가 주어진다.**
- **dnn : 교차표의 행변수, 열변수 등에 사용되는 이름**

Break Time

2.3.3 벡터 내 특정 요소 선택하기

Contents

한 개 만 추출 하는 경우

R 코드

```
39
40  # 2.3.3 벡터 내 특정 요소 선택하기
41  t_vector <- c(11,12,13,14,15,16,17,18,19,20)
42  t_vector
43  t_vector[3]
44
```

t_vector의 세번째 위치의 값을 반환하겠다!

- 함수는 소괄호 ()
- 인덱싱은 대괄호 []

Run(실행)

```
> t_vector <- c(11,12,13,14,15,16,17,18,19,20)
> t_vector
 [1] 11 12 13 14 15 16 17 18 19 20
> t_vector[3]
[1] 13
>
```

t_vector에서 3번째에 위치한 '13' 이 출력됨.

2.3.3 벡터 내 특정 요소 선택하기

Contents

여러 개 추출 하는 경우

R 코드

```
45
46  #여러개 추출하고 싶을 때
47  idx <- c(1,3,5,6)
48  t_vector[idx]
49
50  t_vector[c(1,3,5,6)]
51  t_vector[c(6,5,3,1)] #적은 순서대로 나옴
52
```

c( ) 안에 적은 숫자 순서가 출력되는 값의 순서가 됨.

Run(실행)

```
> t_vector
 [1] 11 12 13 14 15 16 17 18 19 20
```

```
> #여러개 추출하고 싶을 때
> idx <- c(1,3,5,6)
> t_vector[idx]
[1] 11 13 15 16
>
> t_vector[c(1,3,5,6)]
[1] 11 13 15 16
> t_vector[c(6,5,3,1)] #적은 순서대로 나옴
[1] 16 15 13 11
```

인덱싱 번호는 같지만 순서가 반대로 입력되었음.
서로 반대로 출력됨

2.3.3 벡터 내 특정 요소 선택하기

Contents

연속으로 추출 하는 경우

R 코드

```
55  #연속적으로 여러개 추출
56  seq_vector <- 11:20
57  seq_vector
58  t_vector
```

연속적인 숫자를 원할 때는 : 쓰기

```
60  seq_vector <- 51:100
61  seq_vector
62  seq_vector[30:40]
63
```

seq_vector의 30번째~40번째 값 추출

Run(실행)

```
> #연속적으로 여러개 추출
> seq_vector <- 11:20
> seq_vector
 [1] 11 12 13 14 15 16 17 18 19 20
> t_vector
 [1] 11 12 13 14 15 16 17 18 19 20
```

```
> seq_vector <- 51:100
> seq_vector
 [1]  51  52  53  54  55  56  57  58  59  60  61
[12]  62  63  64  65  66  67  68  69  70  71  72
[23]  73  74  75  76  77  78  79  80  81  82  83
[34]  84  85  86  87  88  89  90  91  92  93  94
[45]  95  96  97  98  99 100
> seq_vector[30:40]
 [1] 80 81 82 83 84 85 86 87 88 89 90
```

11:20 과 *c(11,12,....,19,20)* 결과 같음.

2.3.3 벡터 내 특정 요소 선택하기

Contents

Contents

Contents

Contents

Contents

Contents

Contents

논리연산자로 추출하는 경우

R 코드

```r
# 논리 연산자로 추출
log_vector <- 11:15
log_idx <- c(F,F,T,F,F)

log_vector[log_idx] #true 인것만 뽑아냄

log_vector[log_idx==F]
log_vector[!log_idx]
```

논리연산자: TRUE / FALSE
- T 또는 F로 표기해도 됨(단, 대문자!)

인덱싱을 **논리연산자**로 할 경우
- 'TRUE'인 경우만 추출 함
- 반대로 하고 싶다면 ! 사용

Run(실행)

```
> log_vector <- 11:15
> log_idx <- c(F,F,T,F,F)
> log_vector[log_idx] #true 인것만 뽑아냄
[1] 13
> log_vector[log_idx==F]
[1] 11 12 14 15
> log_vector[!log_idx]
[1] 11 12 14 15
```

11~15에서 세번째에 있는 13 추출

! 를 사용하여 T/F가 반대가 됨.

2.3.3 벡터 내 특정 요소 선택하기

논리연산자로 추출 응용

R 코드

```
84  # 응용
85  log_vector > 13
86
87  log_vector[log_vector>13] #T인것 만 뽑아냄
88  log_vector[log_vector>=13]
89  log_vector[log_vector<=13]
90
```

- 부등호 입력하면 논리 연산자로 출력됨
- T인 값만 추출하는 원리

Run(실행)

```
> log_vector > 13
[1] FALSE FALSE FALSE  TRUE  TRUE
>
> log_vector[log_vector>13] #T인것 만 뽑아냄
[1] 14 15
> log_vector[log_vector>=13] #부등호 쓰는 방법
[1] 13 14 15
> log_vector[log_vector<=13]
[1] 11 12 13
```

2.3.3 벡터 내 특정 요소 선택하기

Contents

seq( ) & rep( ) 함수

R 코드

```
67  #seq & rep
68  ?seq
69  seq(from=10,to=20,by=2)
70  seq(10,20,2)
71  seq(20,10,-2)
72
73  rep(0,10)
74  rep(c(0,1),10)
```

함수 디폴트 값 순서로 입력하는 경우 앞에 옵션 입력 안 해도 됨.

- seq (a, b, c)

으로 값 생성

Run(실행)

```
> ?seq
> seq(from=10,to=20,by=2)
[1] 10 12 14 16 18 20
> seq(10,20,2)
[1] 10 12 14 16 18 20
> seq(20,10,-2)
[1] 20 18 16 14 12 10
>
> rep(0,10)
 [1] 0 0 0 0 0 0 0 0 0 0
> rep(c(0,1),10)
 [1] 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
```

by=음수 입력하면 설정 간격으로 작아짐.

Files | Plots | Packages | Help | Viewer

R: Sequence Generation — Find in Topic

seq {base} R Documentation

Sequence Generation

Description

Generate regular sequences. `seq` is a standard generic with a default method. `seq.int` is a primitive which can be much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.

Usage

```
seq(...)

## Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(leng
    length.out = NULL, along.with = NULL, ...
```

2.3.3 벡터 내 특정 요소 선택하기

Contents

which( ) 함수

R 코드

```
> t_vector
 [1] 11 12 13 14 15 16 17 18 19 20
```

```
85  #which 함수
86  which(t_vector==11)
87  t_vector[1]
88
89  wh_vec <- rep(c(0:5),10)
90  which(wh_vec==4)
91
```

크기가 큰 데이터에서 유용

- 찾고자 하는 값이 있는 위치 반환

t_vector에서 11이 있는 위치는?

'같다'라는 의미는 == 로 씀

Run(실행)

```
> #which 함수
> which(t_vector==11)
[1] 1
> t_vector[1]
[1] 11
>
> wh_vec <- rep(c(0:5),10)
> which(wh_vec==4)
 [1]  5 11 17 23 29 35 41 47 53 59
```

값이 1이라는 것이 아니라 11이 있는 **"위치"**가 1이라는 의미

2.3.4 벡터 요소 수정/추가/삭제

Contents

Contents

Contents

Contents

Contents

Contents

Contents

벡터 요소 수정

R 코드

```
92  #데이터 수정
93  m_vector <- 1:5
94
95  m_vector[3] <- 999
96  m_vector
97
98  m_vector <- 1:5
99  m_vector[seq(1,5,2)] <- 0 #홀수는 0으로
100 m_vector[seq(2,5,2)] <- 1 #짝수는 1로
101 m_vector
```

m_vector의 세 번째의 값을 999로 바꾸겠다!

홀수/짝수 위치를 seq() 이용하여 지정

Run(실행)

```
> #데이터 수정
> m_vector <- 1:5
>
> m_vector[3] <- 999
> m_vector
[1]   1   2 999   4   5
>
> m_vector <- 1:5
> m_vector[seq(1,5,2)] <- 0 #홀수는 0으로
> m_vector[seq(2,5,2)] <- 1 #짝수는 1로
> m_vector
[1] 0 1 0 1 0
```

3이 999로 바뀜.

2.3.4 벡터 요소 수정/추가/삭제

벡터 요소 수정

R 코드

```
103  m_vector <- 1:5
104  m_vector[3] <- 10
105  m_vector
106  m_vector <- 10
107  m_vector
108
109  m_vector <- 1:5
110  length(m_vector)
111  m_vector[1:length(m_vector)] <- 10
```

[ ] 없으면 새로운 벡터가 만들어짐.

- 10이 있는 새로운 벡터 생성

벡터 전체를 바꾸려면 처음부터 끝까지의 위치 입력해주어야 함.

Run(실행)

```
> m_vector <- 1:5
> m_vector[3] <- 10
> m_vector
[1]  1  2 10  4  5
> m_vector <- 10
> m_vector
[1] 10
>
> m_vector <- 1:5
> length(m_vector)
[1] 5
> m_vector[1:length(m_vector)] <- 10
```

length( )는 전체 데이터의 **길이(개수)**를 나타냄.

2.3.4 벡터 요소 수정/추가/삭제

Contents

벡터 요소 추가

R 코드

```
113  #추가
114  m_vector <- 1:5
115  add_vector <- c(999,m_vector);add_vector
116  add_vector <- c(m_vector,999);add_vector
117
118  add_vector <- c(m_vector,100:102);add_vector
119  add_vector <- c(100:102,m_vector);add_vector
120
121  c(m_vector, add_vector)
```

] 입력하는 순서가 출력 순서

벡터와 벡터의 결합
c (벡터1,벡터2...)

Run(실행)

```
> #추가
> m_vector <- 1:5
> add_vector <- c(999,m_vector);add_vector
[1] 999   1   2   3   4   5
> add_vector <- c(m_vector,999);add_vector
[1]   1   2   3   4   5 999
> add_vector <- c(m_vector,100:102);add_vector
[1]   1   2   3   4   5 100 101 102
> add_vector <- c(100:102,m_vector);add_vector
[1] 100 101 102   1   2   3   4   5
> c(m_vector, add_vector)
 [1]   1   2   3   4   5 100 101 102   1   2   3   4   5
```

2.3.4 벡터 요소 수정/추가/삭제

Contents

원하는 위치에 벡터 추가

R 코드

```
123  #append 함수
124  vec_a <- c("가","나","마","바")
125  vec_b <- c("다","라")
126  vec_a; vec_b
127  append(vec_a,vec_b,2)
128
```

- append(a,b,c)
  - b를 a의 c번째 다음에 결합

vec_b를 vec_a의 **3번째 위치** 부터 추가

Run(실행)

```
> #append 함수
> vec_a <- c("가","나","마","바")
> vec_b <- c("다","라")
> vec_a; vec_b
[1] "가" "나" "마" "바"
[1] "다" "라"
> append(vec_a,vec_b,2)
[1] "가" "나" "다" "라" "마" "바"
```

2.3.4 벡터 요소 수정/추가/삭제

Contents

Contents

Contents

Contents

Contents

Contents

Contents

벡터 요소 제거

R 코드

```
129  #제거하기
130  t_vector <- 11:20
131  t_vector[c(1,3,5,6)]
132  t_vector[-c(1,3,5,6)]
133  t_vector[-length(t_vector)]
134
135  #논리형 일때
136  log_var <- c(F,T,T,F,F)
137  t_vector[log_var]
```

"특정 값 제거하고 싶을 때는 - 사용"

특정 값 추출하고 싶을 때

특정 값 제거하고 싶을 때

Run(실행)

```
> t_vector <- 11:20
> t_vector[c(1,3,5,6)]
[1] 11 13 15 16
> t_vector[-c(1,3,5,6)]
[1] 12 14 17 18 19 20
> t_vector[-length(t_vector)]
[1] 11 12 13 14 15 16 17 18 19
> #논리형 일때
> log_var <- c(F,T,T,F,F)
> t_vector[log_var]
[1] 12 13 17 18
```

벡터 개수 만큼 입력되지 않으면 자동으로 채움.

- 벡터 수가 10개, 논리연산자는 5개
- log_var 두번 자동 반복

2.3.5 벡터의 연산

Contents

Contents

Contents

Contents

Contents

Contents

Contents

논리연산자로 추출하는 경우

R 코드

```
139  #벡터의 연산
140  a <- c(1,2,3,4); b <- c(5,6,7,8)
141  a+b; a*b
142
143  b <- 2:3; a+b
144
145  b <- 2:4; a+b
```

a와 값 개수가 다르지만 배수이기 때문에 4개 **자동 맞춤**

배수가 아니라서 에러 뜨지만 그래도 계산은 해줌.

Run(실행)

```
> a <- c(1,2,3,4); b <- c(5,6,7,8)
> a+b; a*b
[1]  6  8 10 12
[1]  5 12 21 32
>
> b <- 2:3; a+b
[1] 3 5 5 7
>
> b <- 2:4; a+b
[1] 3 5 7 6
Warning message:
In a + b : longer object length is not a multiple of shorter object length
```

c(2,3,2,3)으로 자동 채우기

c(2,3,4,2)으로 자동 채우기

5.6.2 샘플링

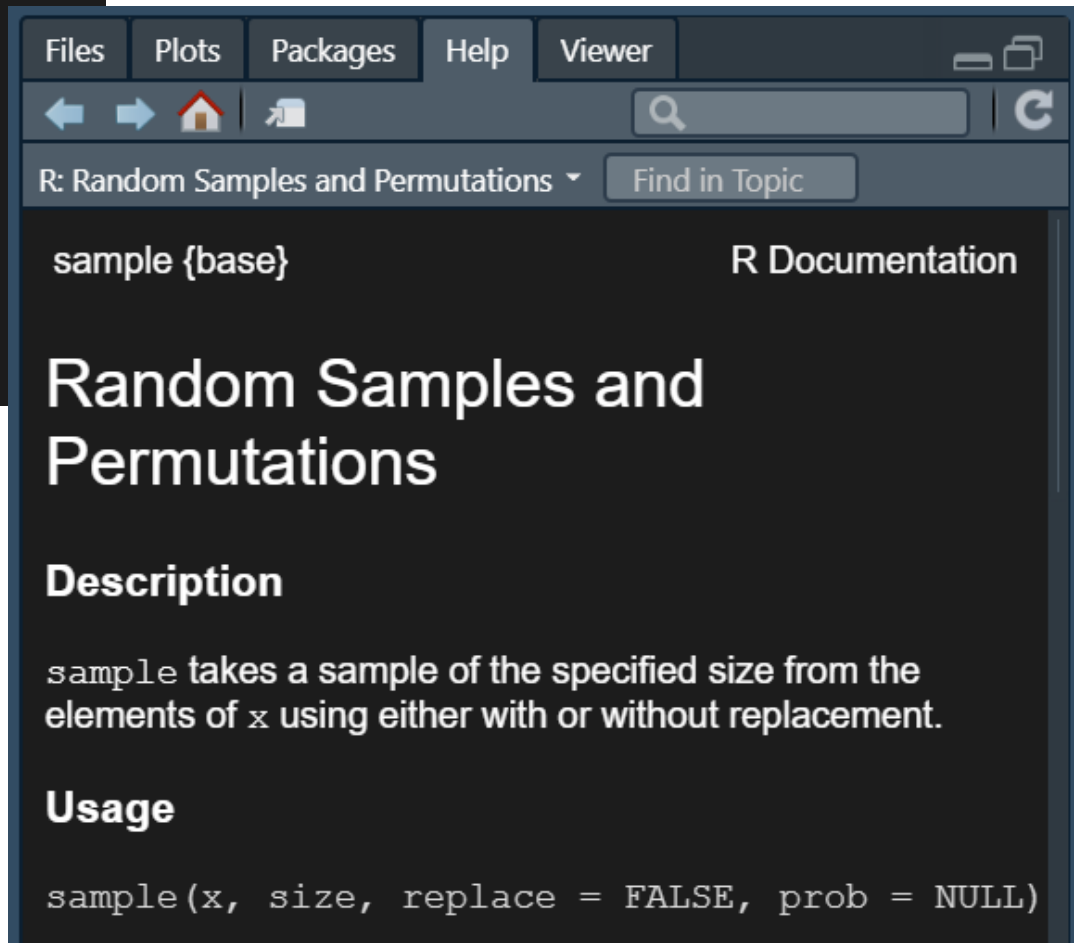
Contents

sample( ) 함수

1부터 6 중에서 무작위로 1개 뽑기

R 코드

```
130  #샘플링
131  ?sample
132  dice <- sample(c(1:6),1); dice
133  sample(c(1:6),7,T)
134
135  sample(c(1:10),2)
136  sample(c(1:10),2)
137  sample(c(1:10),2)
```



- **복원 추출(replace=T)**
  - 뽑은 것을 다시 넣고 뽑는다

- **비복원 추출(replace=F)**
  - 한 번 뽑은 것은 다시 뽑지 않는다

5.6.2 샘플링

Contents

Contents

Contents

Contents

Contents

Contents

sample( ) 함수

Run(실행)

```
> ?sample
> dice <- sample(c(1:6),1); dice
[1] 2
> sample(c(1:6),7,T)
[1] 6 2 2 2 1 2 1
>
> sample(c(1:10),2)
[1] 3 2
> sample(c(1:10),2)
[1]  7 10
> sample(c(1:10),2)
[1] 7 4
```

복원 추출

-> 같은 수 여러 번 뽑힘.

샘플을 뽑을 때마다 값이 달라짐.

- 같은 코드에 같은 값이 나오려면 **set.seed( )** 사용해야 함.

5.6.2 샘플링

Contents

Contents

Contents

Contents

Contents

Contents

set.seed( ) 함수

R 코드

```
139  set.seed(0804)
140  sample(c(1:10),2)
141  set.seed(0804)
142  sample(c(1:10),2)
143
144  set.seed(0921)
145  coin <- sample(c("앞","뒤"),100,replace = T)
146  table(coin)
147  coin <- sample(c("앞","뒤"),100,replace = T)
148  table(coin)
149  coin <- sample(c("앞","뒤"),100,replace = T)
150  table(coin)
151  coin <- sample(c("앞","뒤"),100,replace = T)
152  table(coin)
153
```

- **set.seed(a)**
  - a는 '시드넘버'
  - 원하는 임의 수 지정

시드넘버가 같으면 출력값도 같음

- 사용법
  - 시드 넘버 지정하고 싶은 부분을 전체 블록 처리하여 실행

```
138
139  set.seed(0804)
140  sample(c(1:10),2)
141  set.seed(0804)
```

5.6.2 샘플링

Contents

Contents

Contents

Contents

Contents

Contents

set.seed( ) 함수

Run(실행)

```
> set.seed(0804)
> sample(c(1:10),2)
[1] 9 6
> set.seed(0804)
> sample(c(1:10),2)
[1] 9 6
```

시드넘버 (0804)로 같게 입력

- 같은 값 샘플링 됨

동전 뒤집기(응용)

```
> set.seed(0921)
> coin <- sample(c("앞","뒤"),100,replace = T); table(coin)
coin
뒤 앞
49 51
> coin <- sample(c("앞","뒤"),100,replace = T); table(coin)
coin
뒤 앞
51 49
> coin <- sample(c("앞","뒤"),100,replace = T); table(coin)
coin
뒤 앞
49 51
> coin <- sample(c("앞","뒤"),100,replace = T); table(coin)
coin
뒤 앞
52 48
```

=> 앞, 뒤의 비율이 비슷함을 확인 할 수 있음.

5.6.3 결측값 대체하기

Contents

Contents

Contents

Contents

Contents

Contents

결측값 처리

R 코드

* 결측값은 대부분 **NA**로 입력하거나 입력 되어있음.

```r
#결측값
na.vector <- c(1,2,NA,4,5,NA)
is.na(na.vector)
sum(is.na(na.vector))

sum(na.vector)
omit.vector <- na.omit(na.vector)
sum(omit.vector)
sum(na.vector, na.rm = T)

na.vector[is.na(na.vector)] <- 999
na.vector

```

- **is.na(a)**
  - a 안에 결측값이 있는지 논리형 (T/F) 으로 출력
- **sum(is.na(a))**
  - a의 결측치 개수 확인
- **na.omit(a)**
  - a의 결측치 제거
- **na.rm=T**
  - 함수 내의 옵션
  - 결측치 빼고 계산

5.6.3 결측값 대체하기

Contents

Contents

Contents

Contents

Contents

Contents

결측값 처리

Run(실행)

```
> #결측값
> na.vector <- c(1,2,NA,4,5,NA)
> is.na(na.vector)
[1] FALSE FALSE  TRUE FALSE FALSE  TRUE
> sum(is.na(na.vector))
[1] 2
>
> sum(na.vector)
[1] NA
> omit.vector <- na.omit(na.vector)
> sum(omit.vector)
[1] 12
> sum(na.vector, na.rm = T)
[1] 12
>
> na.vector[is.na(na.vector)] <- 999
> na.vector
[1]   1   2 999   4   5 999
```

논리형에서 T는 1로 F는 0으로 계산됨.

데이터 안에 결측치가 있을 경우 계산 안됨 -> **두가지** 방법

1. na.omit 함수 사용
2. 함수 내의 na.rm=T 옵션 사용

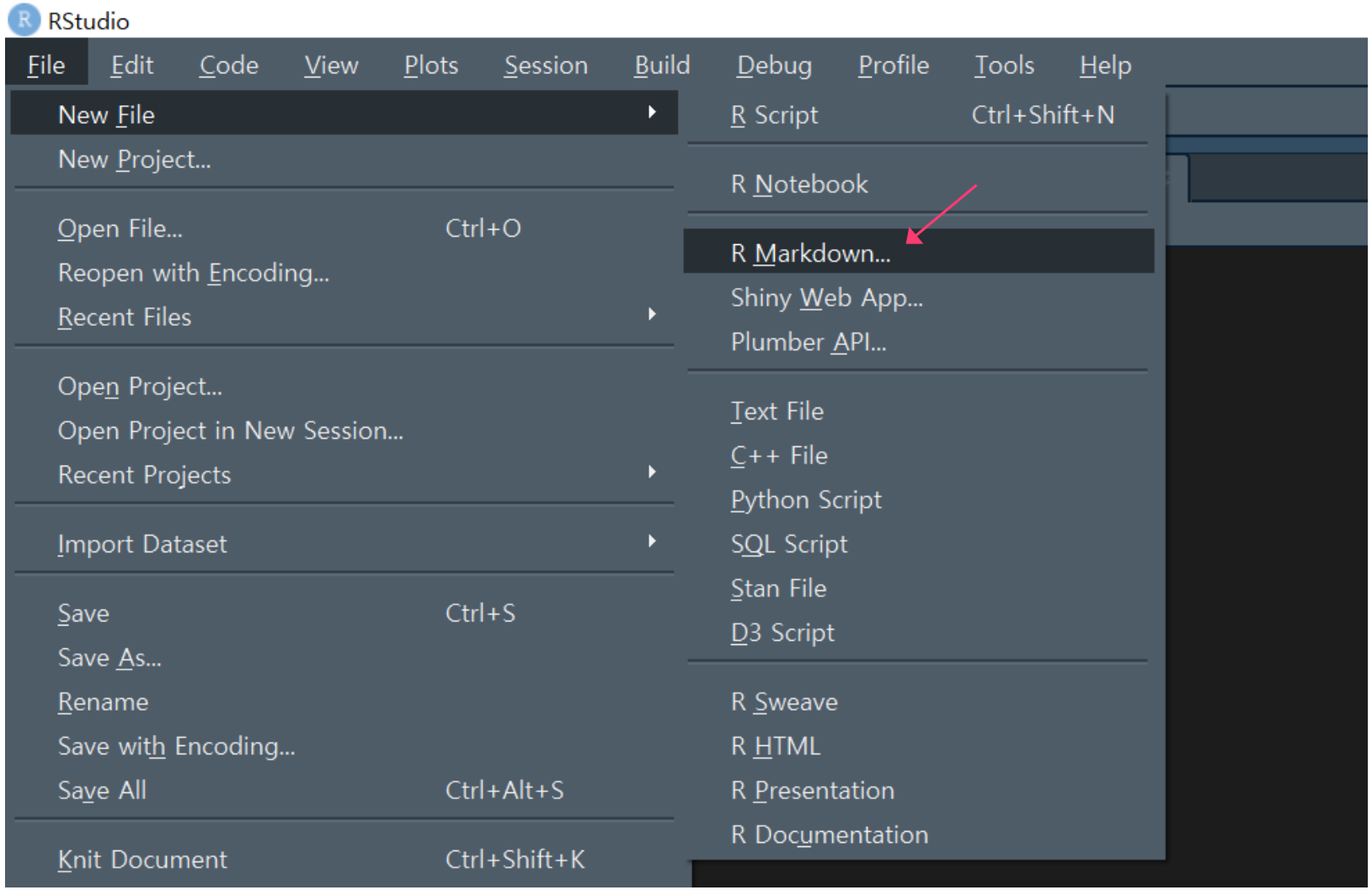
0. R 마크다운 이란?

R에서 문서를 간편하게 작성할 수 있는 기능

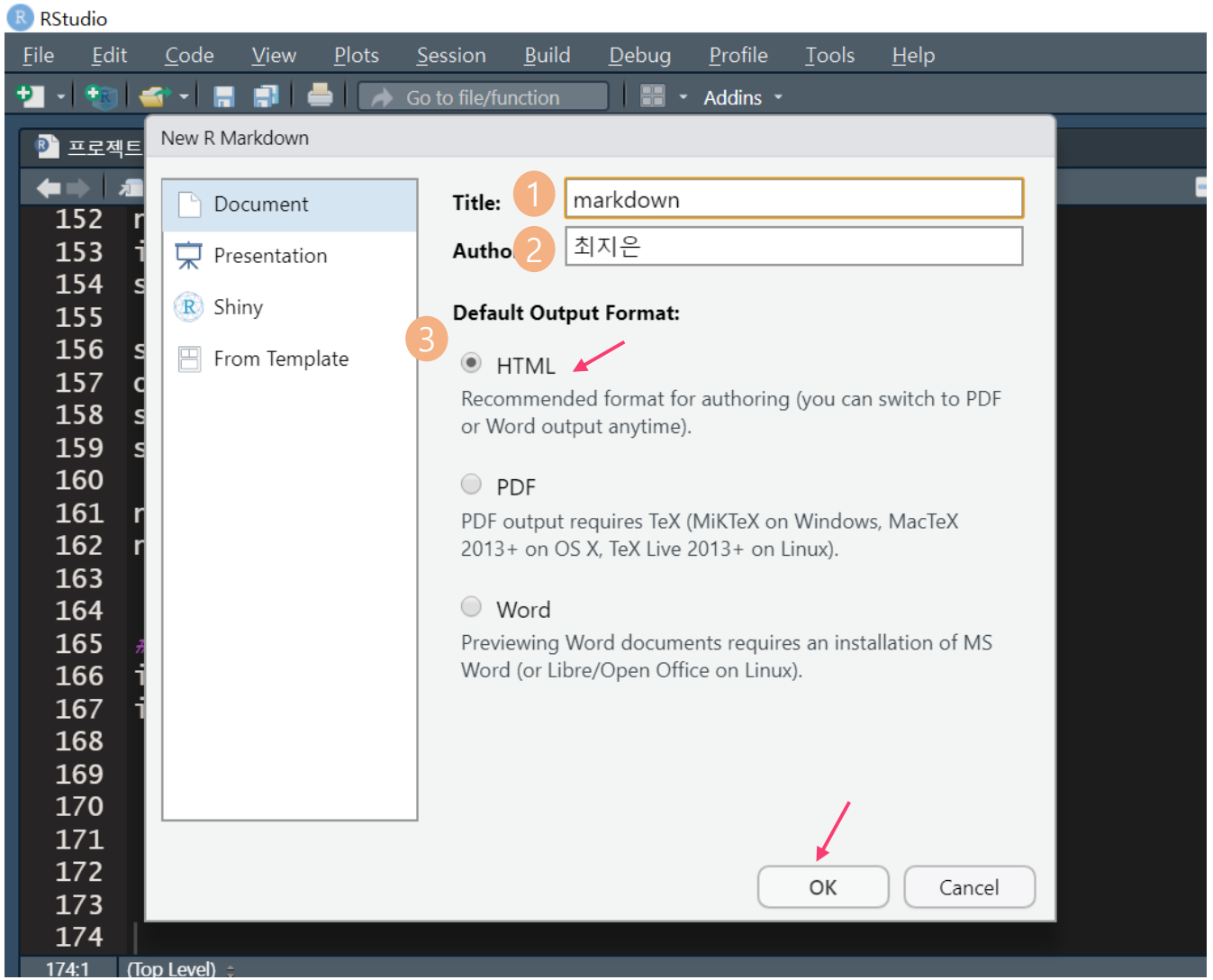
1. 마크다운 패키지 설치

```
165  #R markdown
166  install.packages("rmarkdown")
167  install.packages("knitr")
168
```

2. 마크다운 실행하기(File -> New File -> R Markdown)



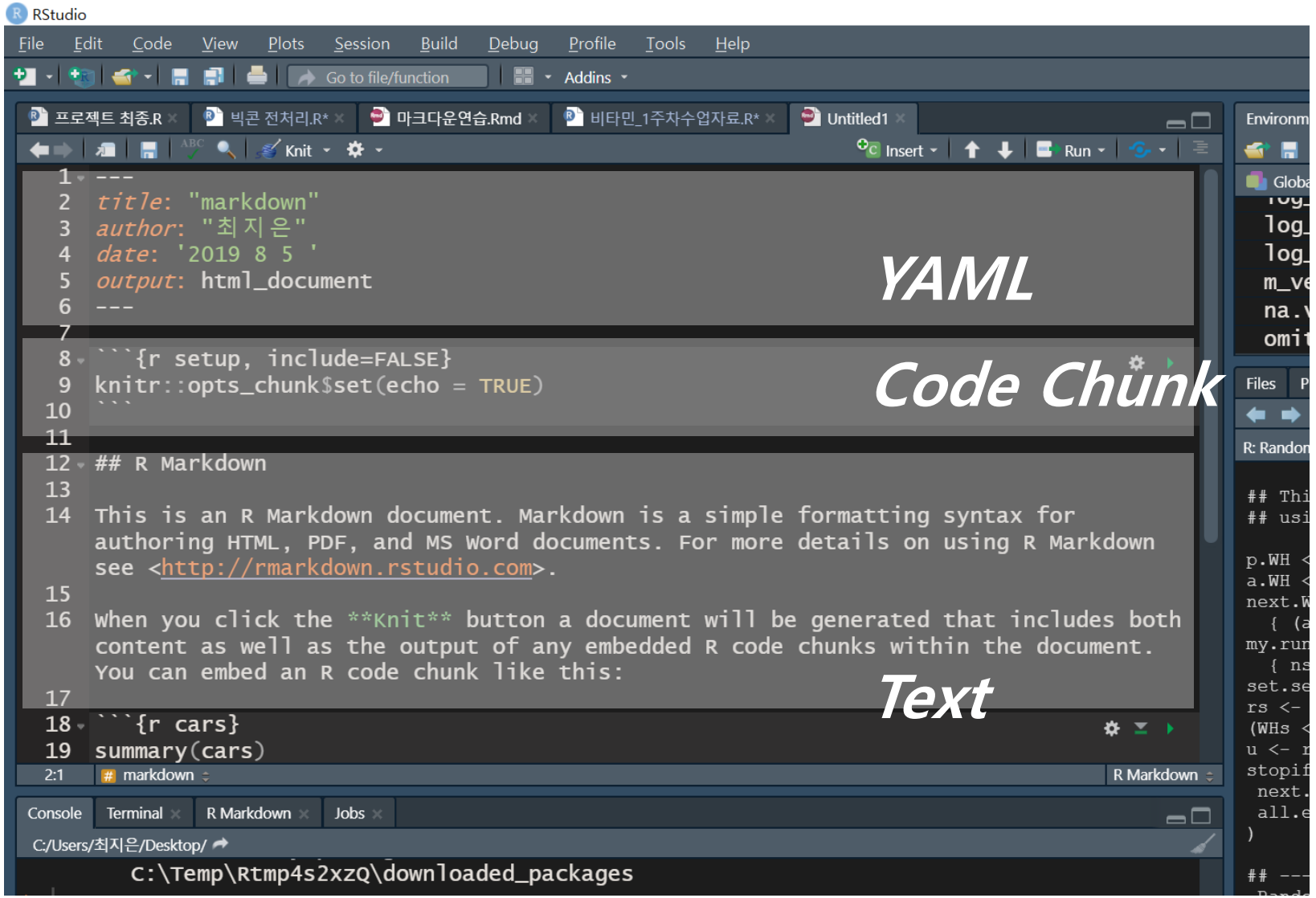
2. 마크다운 실행하기(File -> New File -> R Markdown)



1. 원하는 문서 제목
2. 작성자 이름
3. 원하는 문서 형식
   - html
   - word
   - pdf

다 지정 했으면 OK 누르기

3. R 마크다운 구조 설명

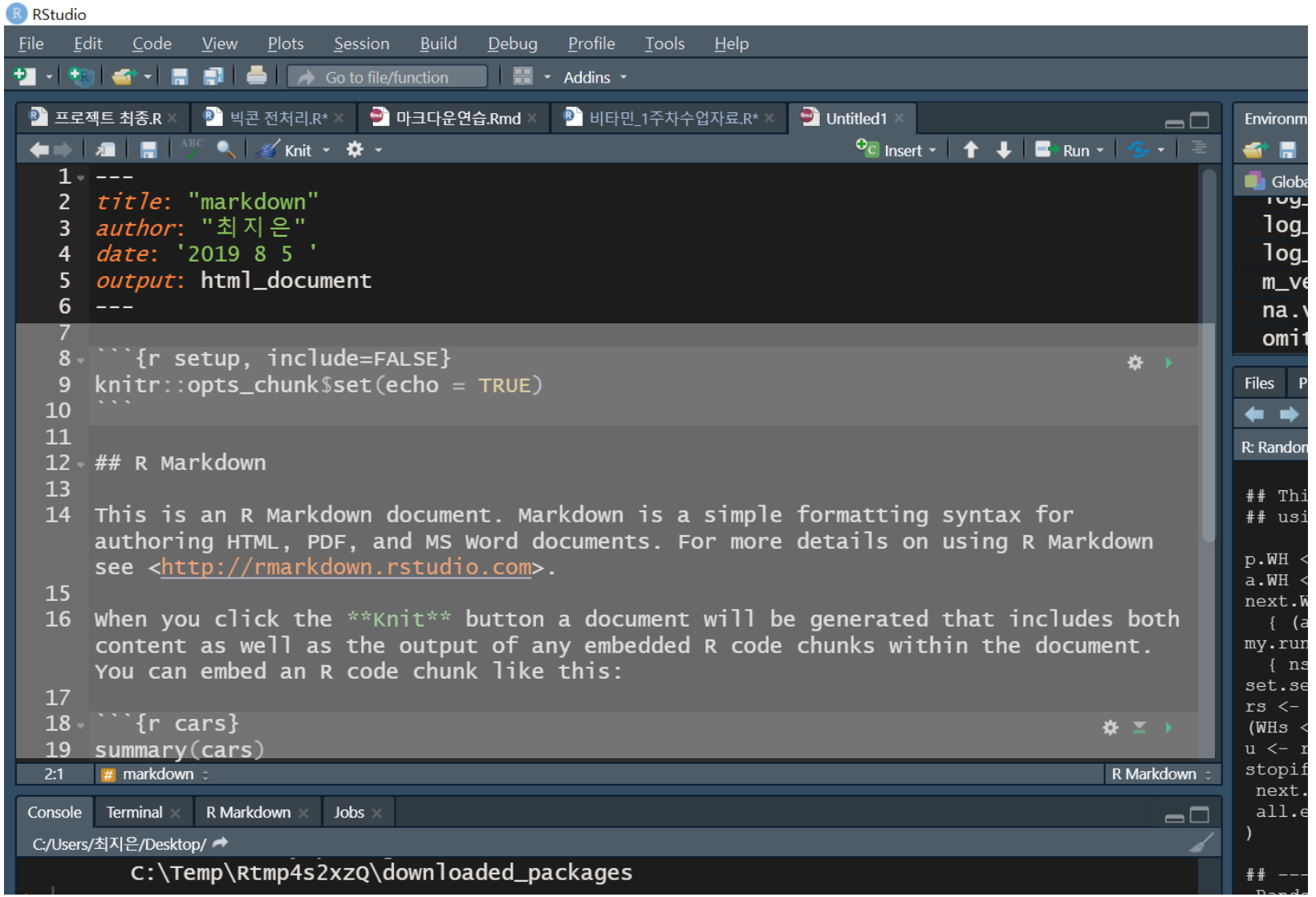


1. **YAML**
 - 문서의 기본적인 정보 (설정할 필요 없음)

2. **Code Chunk**
 - r 코드 작성하는 부분

3. **Text**
 - 부가적인 설명 작성

3. R 마크다운 구조 설명



처음에 뜨는 이 부분은 markdown 예시.

markdown 문법 어려울 때 참고하기 좋음.

3. R 마크다운 구조 설명

실행 후 웹페이지(html)에서 보여지는 형태

Untitled

최지은

2019 8 6

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

입력했던 R 코드

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

R 코드 실행결과

4. 기초적인 마크다운 문법

4.1 R 코드 삽입

```
17  t_vector <- c(11,12,13,14,15,16,17,18,19,20)
18  t_vector
19  t_vector[3]
20
```

R studio 에서의 작성 방식

````
17
18  ```{r}
19  t_vector <- c(11,12,13,14,15,16,17,18,19,20)
20  t_vector
21  t_vector[3]
22  ```
23     > 세 번째 있는 것 추출하는 코드
24
````

R markdown에서의 작성방식

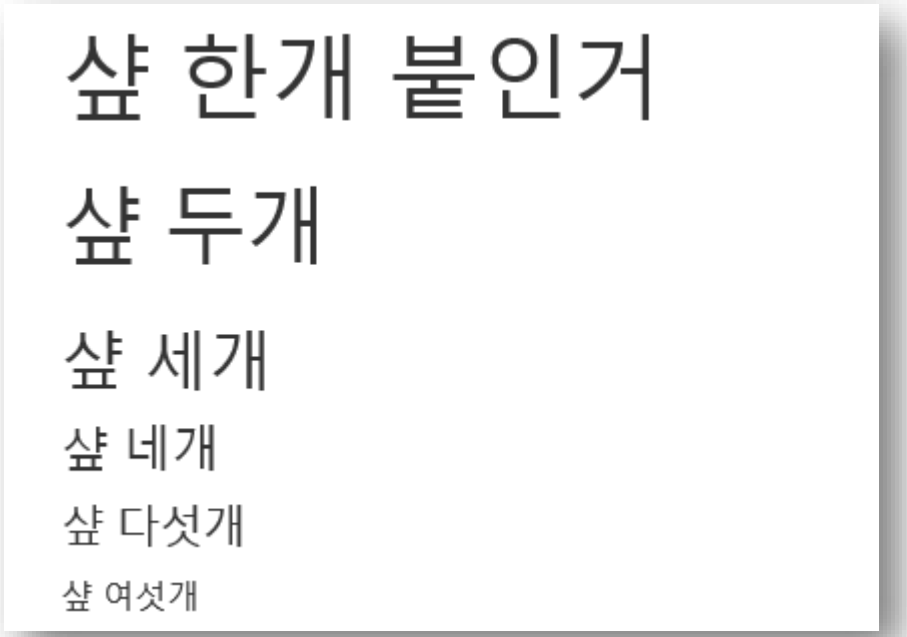
R 코드를 넣을 때는
```{r} 로 시작하고 코드의 마무리는 ```로 지어야함.
```

4. 기초적인 마크다운 문법

4.2 Text 작성 문법

```
# 샾 한개 붙인거
## 샾 두개
### 샾 세개
#### 샾 네개
##### 샾 다섯개
###### 샾 여섯개
```

R 마크다운 작성 문법



실행 후 웹페이지(html)에서 보여지는 형태

글자 크기는 #의 개수로 조절한다
- 제일 큰 글자크기가 #
- 가장 작은 글자크기가 ######

#쓰고 한 칸 띄어야 적용됨!

4. 기초적인 마크다운 문법

4.2 Text 작성 문법

```
286
287 *별표 한개는 이탤리체*
288
289
290 **별표 두개는 볼드체**
291
292
293 ***별표 세개는 볼드 & 이탤리체***
294
295
296 ~~물결 두개는 취소선~~
297
```

R 마크다운 작성 문법

별표 한개는 이탤리체

별표 두개는 볼드체

별표 세개는 볼드 & 이탤리체

~~물결 두개는 취소선~~

실행 후 웹페이지(html)에서 보여지는 형태

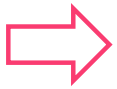
텍스트 강조는 *, ~를 이용하여 나타냄

4. 기초적인 마크다운 문법

4.2 Text 작성 문법

R 마크다운 작성 문법

```
16
17 ```{r}
18 t_vector <- c(11,12,13,14,15,16,17,18,19,20)
19 t_vector
20 t_vector[3]
21 ```
22    > 세 번째 있는 것 추출하는 코드
23
```



```
t_vector <- c(11,12,13,14,15,16,17,18,19,20)
t_vector
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```
t_vector[3]
```

```
## [1] 13
```

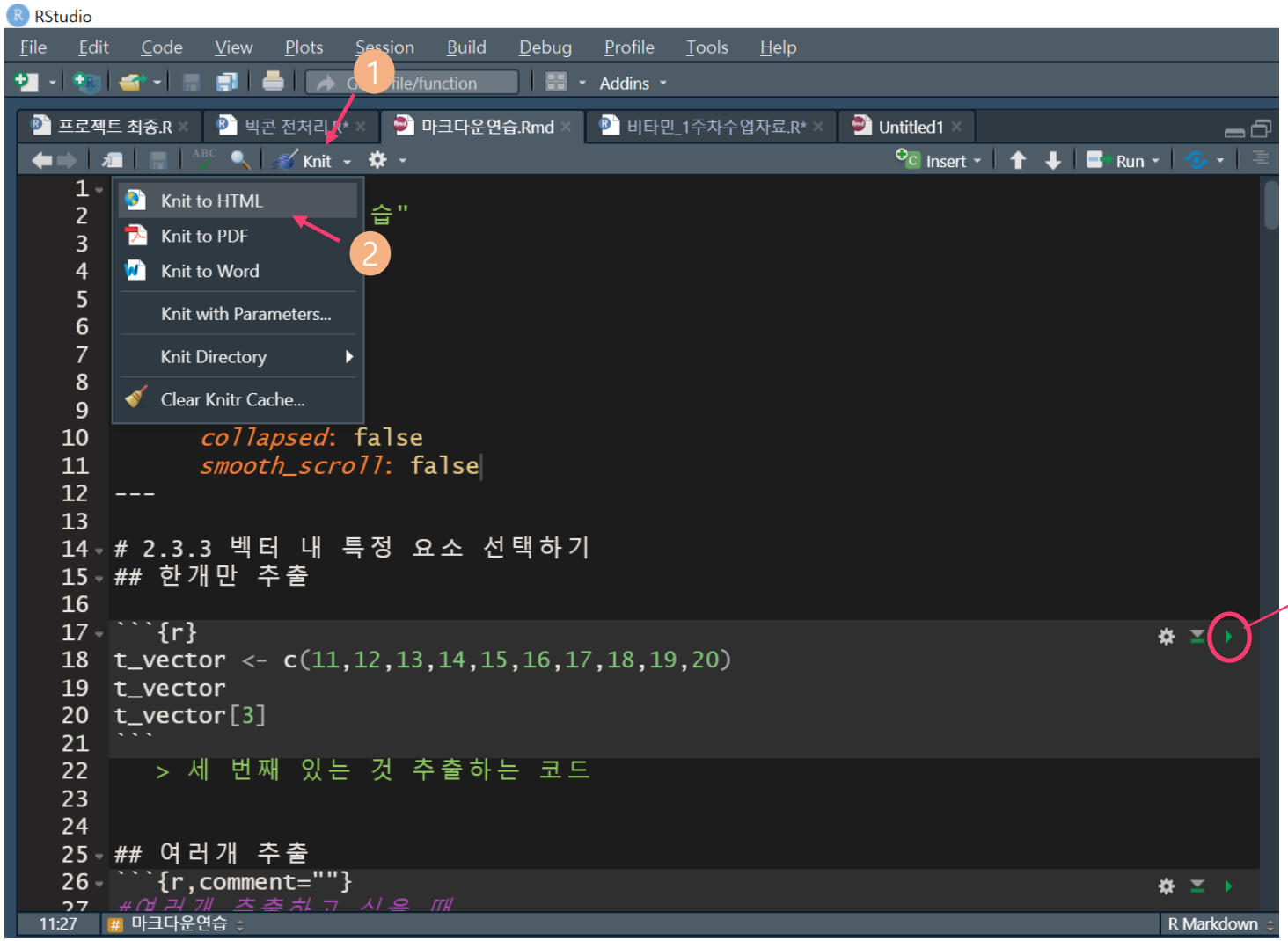
세 번째 있는 것 추출하는 코드

코드나 결과에 대한 설명을 할 때는

> 를 사용하면 됨.

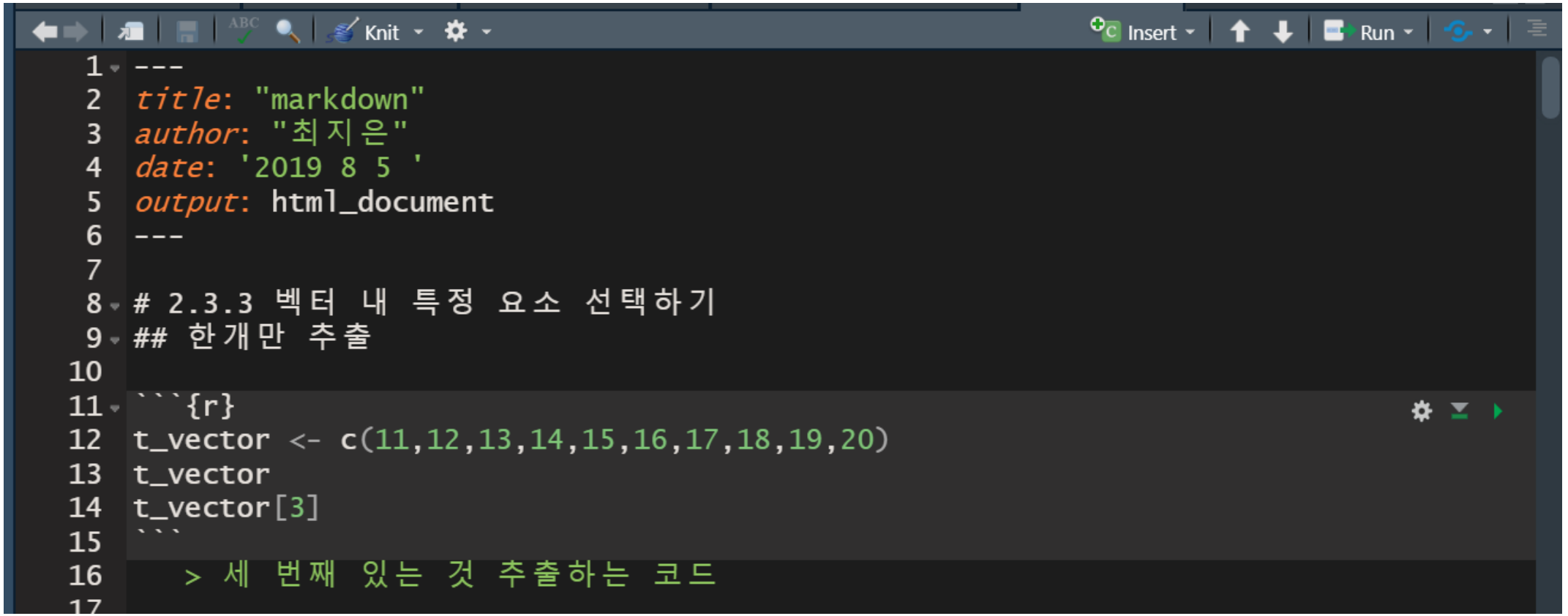
실행 후 웹페이지(html)에서 보여지는 형태

5. 최종 실행방법(knit 클릭 -> knit to HTML 클릭)



R 코드가 실행이 되는지 개별적으로 확인하고 싶으면 이 버튼 누르기

6. 최종 결과물



R 마크다운 작성 문법

6. 최종 결과물

실행 후 웹페이지(html)에서 보여지는 형태

markdown

최지은

2019 8 5

2.3.3 벡터 내 특정 요소 선택하기

한개만 추출

```
t_vector <- c(11,12,13,14,15,16,17,18,19,20)
t_vector
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```
t_vector[3]
```

```
## [1] 13
```

> 세 번째 있는 것 추출하는 코드

1주차

R기초 & 데이터 구조

Q & A

교육부 운영진