R 기반 의학통계 및 머신러닝

박 승

충북대학교
CHUNGBUK NATIONAL UNIVERSITY

CHAPTER

01

R 이해하기

01 R 이해하기

R이란?

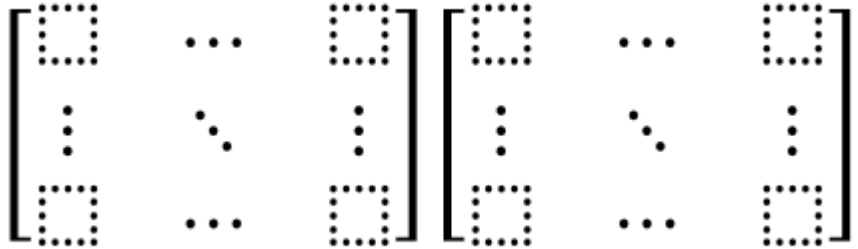
- 통계 분석 및 데이터 분석에 특화된 프로그래밍 언어
  - S언어 기반으로 탄생했으며, 오픈소스로 제공
- 이후 다양한 통계학자와 데이터 과학자들이 개발에 참여하면서, CRAN을 통해 배포
- 방대한 패키지 제공
- 학계 및 연구 기관에서 널리 이용되며, 데이터 분석 및 시각화의 표준 도구 중 하나

이미지 출처 : CRAN

01 R 이해하기

R이란?

- 통계학자 및 데이터 분석가가 직관적으로 사용 가능
- 수학 및 통계 연산을 직관적으로 수행할 수 있도록 다양한 내장 함수 제공
- 데이터를 다루는 것이 핵심이므로, 벡터 연산 및 행렬 연산이 지원됨



이미지 출처 : FLATICON

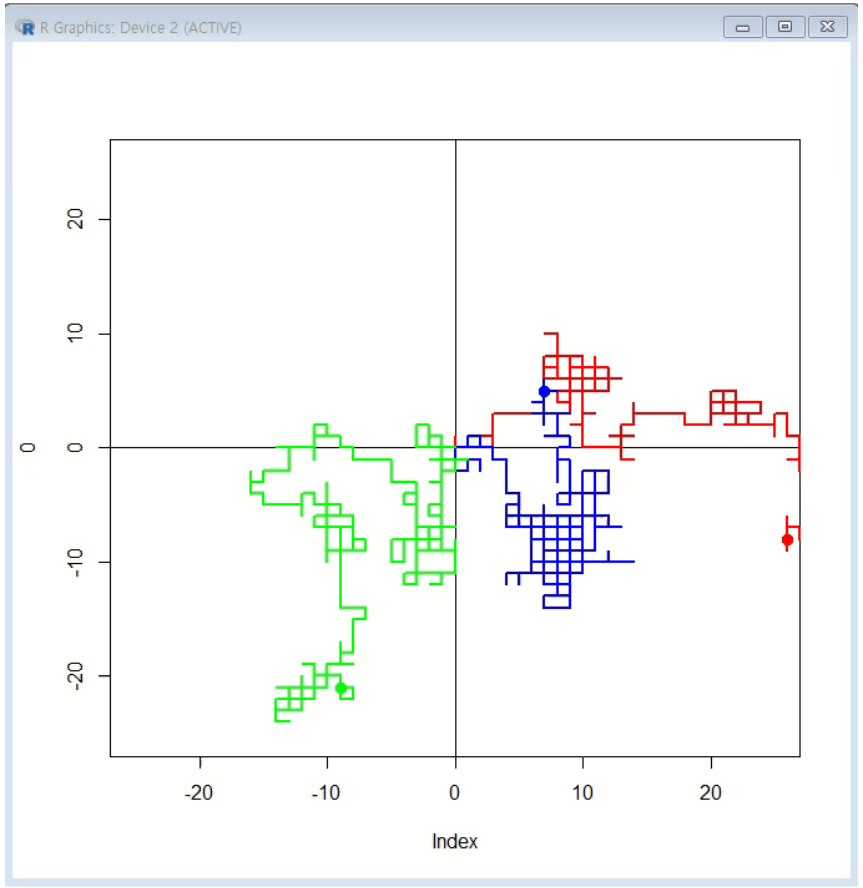
R의 장점

- 1. 통계 분석 및 데이터 분석에 특화된 프로그래밍 언어
  - 통계 분석이 기본 내장되어 있어 별도의 라이브러리 없이도 다양한 분석 가능
  - 회귀 분석, 생존 분석, 분산 분석 등 다양한 빈도주의 및 베이지안 분석 기법 지원
  - lm(), glm(), coxph() 등 직관적인 통계 모델링 함수 제공
  - 줄 단위의 컴파일링을 통해 쉬운 디버깅이 가능하며 복잡한 환경설정이 필요없음

- 2. 오픈 소스 & 강력한 커뮤니티 지원
  - 무료로 사용 가능하며, 전 세계 연구자들이 개발하는 오픈 소스 패키지 사용 가능
  - CRAN(The Comprehensive R Archive Network)에서 수천 개의 패키지 제공
  - Stack Overflow, R-bloggers 등에서 방대한 자료와 Q&A 지원

R의 장점

- 3. 데이터 시각화 지원
  - ggplot2, lattice 등의 단순하지만 강력한 시각화 패키지 제공
  - 논문 및 보고서에 사용할 수 있는 고품질 그래프 제작 가능
  - gganimate 등을 활용한 동적 데이터 시각화 가능



[random walk 시뮬레이션 영상]

R의 장점

- 4. 데이터 처리 및 전처리에 강함
  - tidyverse(dplyr, tidyr, readr 등) 패키지를 활용한 데이터 처리 최적화
  - data.table을 활용하여 대용량 데이터 처리 속도 개선 가능
  - dbplyr, RMySQL 패키지를 활용하여 SQL과의 연동도 용이하게 진행 가능

- 5. 친절한 도움말과 용례 및 풍부한 내장 예제 자료
  - 함수 앞에 물음표(ex) ?hist, ??hist)를 입력하여 함수에 대한 도움말과 용례를 불러올 수 있음
  - 패키지 안에 포함된 예제 자료를 활용하여 스스로 학습이 용이함

R의 단점

- 1. 프로그래밍 문법 자체에 대한 학습이 필요
  - R은 기본적으로 함수형 프로그래밍 언어이며, Python이나 C 계열 언어와 문법이 다름
  - R 특유의 벡터화 연산 개념을 익히는 데 시간이 필요

- 2. 개발 환경으로는 부적합
  - 범용 프로그래밍 언어가 아니므로 웹 개발, 시스템 프로그래밍, 네트워크 프로그래밍에 약함
  - 대규모 소프트웨어 개발에는 Python, Java, C++ 등이 더 적합

R의 단점

- 3. 느린 속도
  - R은 for문 반복 연산이 상대적으로 느리며 벡터화 연산을 적극적으로 활용해야 성능 개선 가능 (메모리 재할당, 실행 시 타입 검사, 바이트코드 컴파일 방식 등의 이유)

- 4. 메모리 사용량이 많음
  - 데이터를 메모리에 적재하는 방식으로, 대용량 데이터 처리에 한계가 있음
  - Python의 pandas 보다 메모리 효율이 낮고, Spark 등과의 연동이 불편

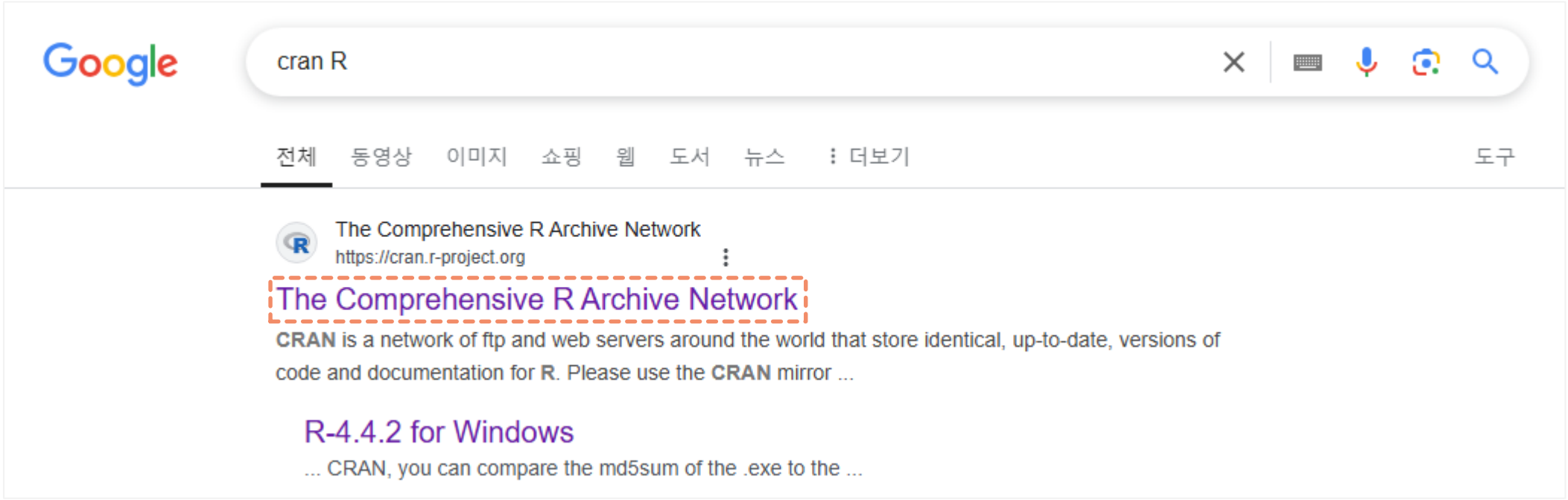
R의 단점

- 5. 딥러닝 및 대규모 머신러닝 생태계 부족
  - 머신러닝 지원 패키지가 많지만, Python(scikit-learn, Tensorflow, Pytorch)에 비해 부족
  - 딥러닝 모델 구축 시 reticulate 연동을 통해 Python을 함께 사용해야 하는 경우가 많음

- 결론
  - R은 통계 및 연구 분석에 강점이 있고, 소프트웨어 개발 및 대규모 데이터 엔지니어링 적용이 어려움
  - R을 필요한 곳에 적절히 사용 하는 것이 필요

01 R 이해하기

R의 설치



R의 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.

R의 설치

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 >= 4.0.x).
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 4.0.x).
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.

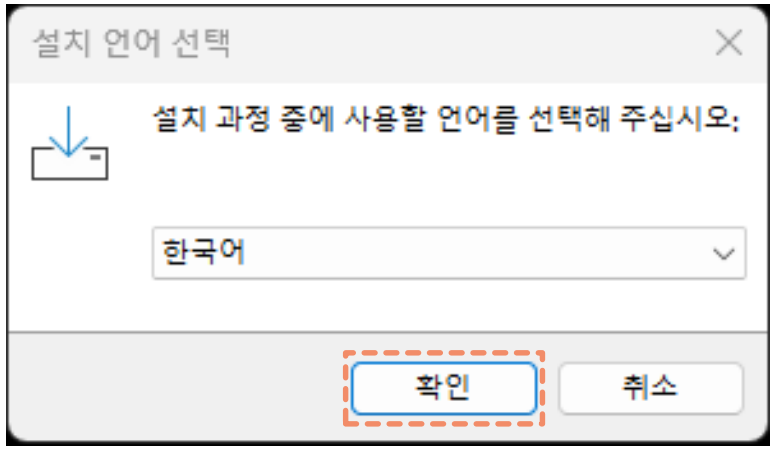
Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

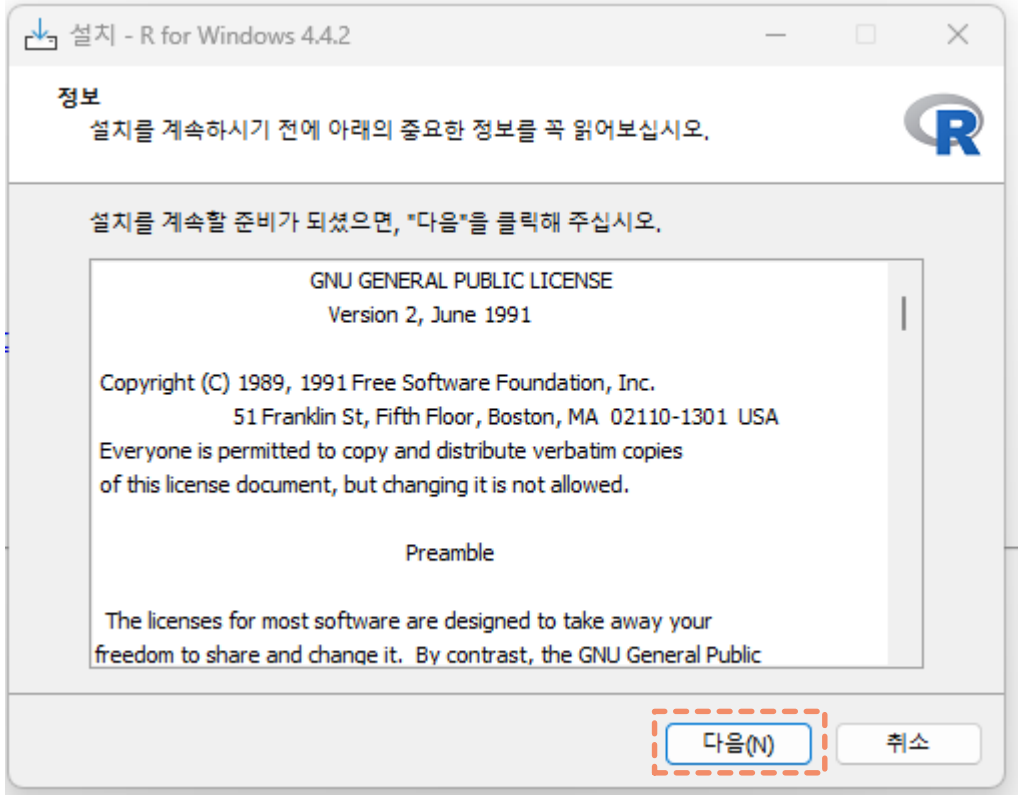
You may also want to read the R FAQ and R for Windows FAQ.

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

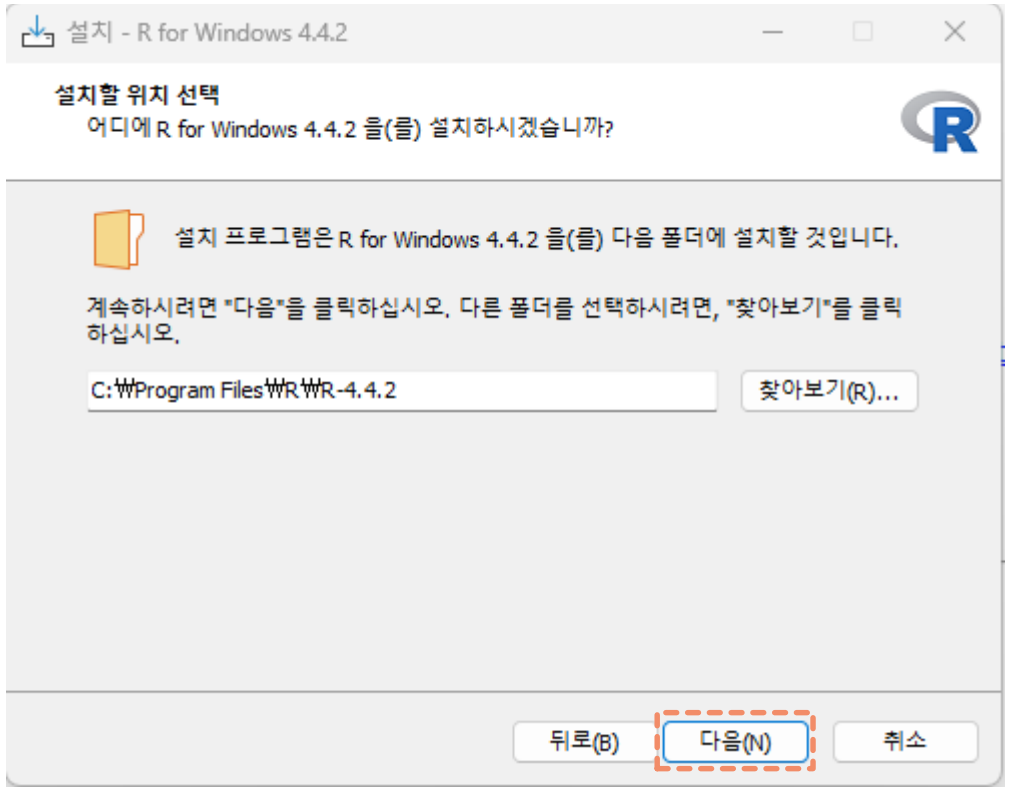
R의 설치

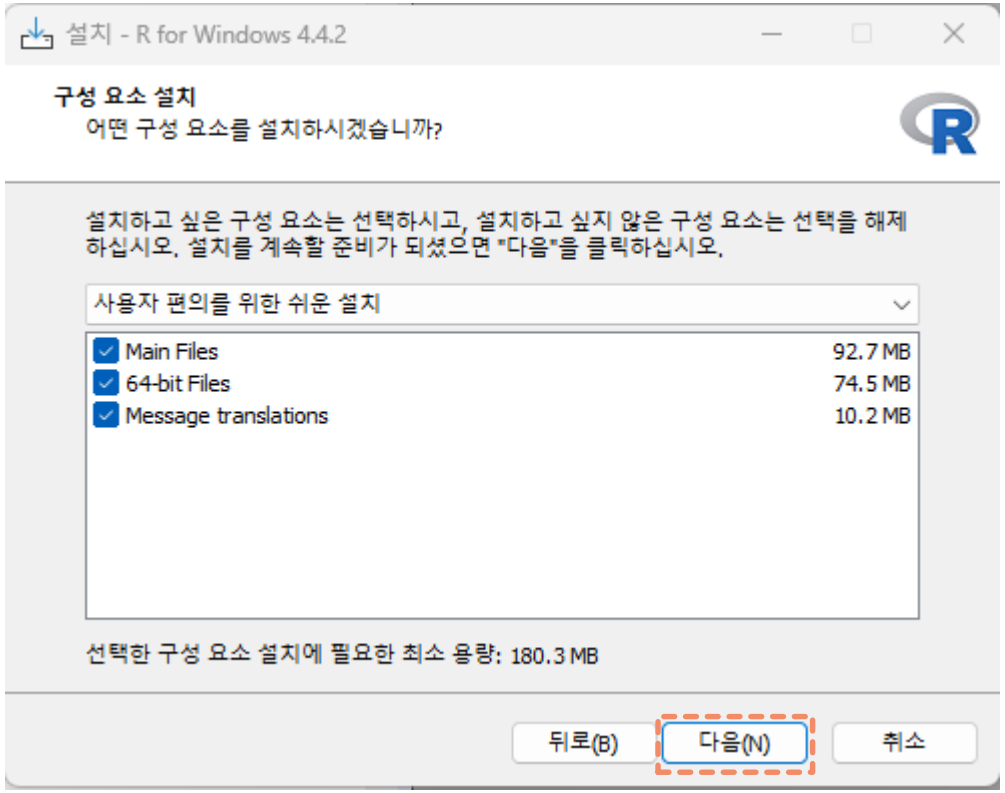

Download R-4.4.2 for Windows (83 megabytes, 64 bit)
README on the Windows binary distribution
New features in this version


설치 언어 선택
설치 과정 중에 사용할 언어를 선택해 주십시오:
한국어
확인
취소

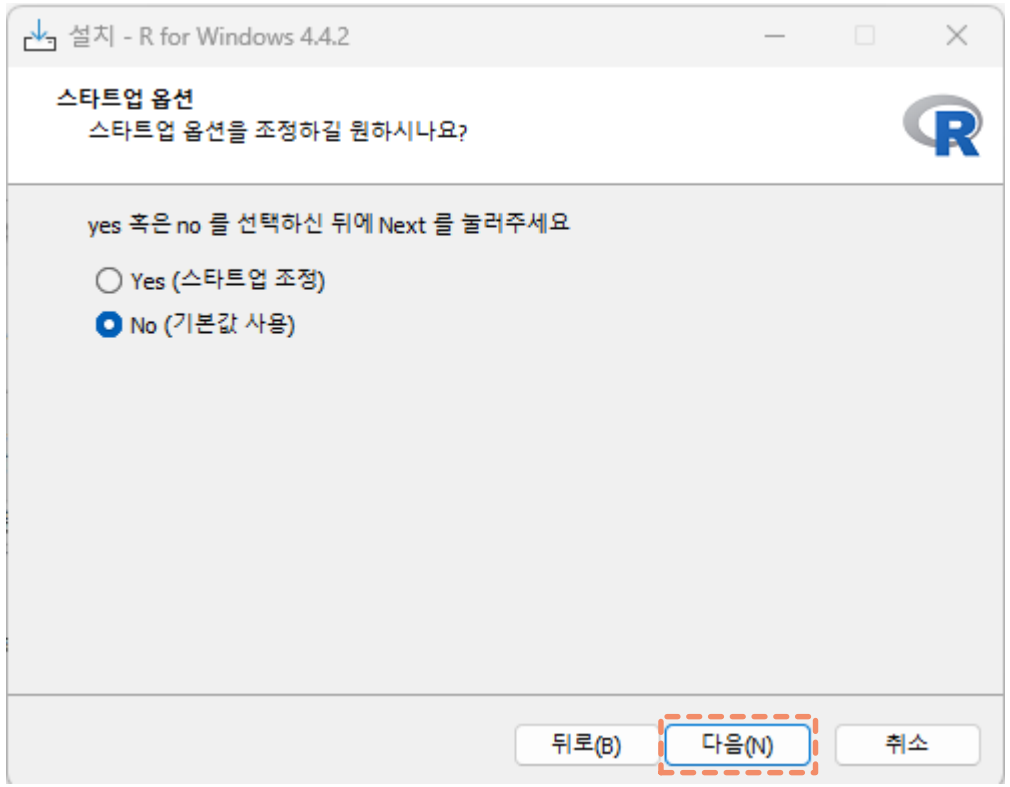

설치 - R for Windows 4.4.2
정보
설치를 계속하시기 전에 아래의 중요한 정보를 꼭 읽어보십시오.
설치를 계속할 준비가 되셨으면, "다음"을 클릭해 주십시오.
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
다음(N)
취소

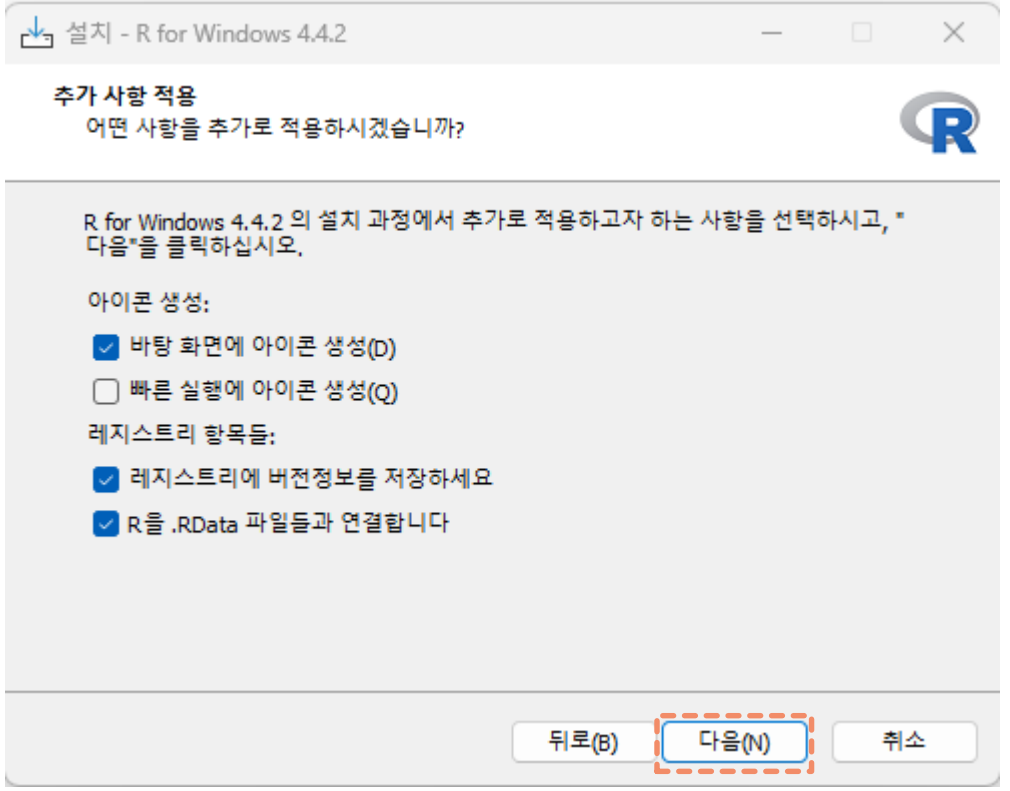
R의 설치


설치 - R for Windows 4.4.2
설치할 위치 선택
어디에 R for Windows 4.4.2 을(를) 설치하시겠습니까?
설치 프로그램은 R for Windows 4.4.2 을(를) 다음 폴더에 설치할 것입니다.
계속하시려면 "다음"을 클릭하십시오. 다른 폴더를 선택하시려면, "찾아보기"를 클릭하십시오.
C:₩Program Files₩R₩R-4.4.2
찾아보기(R)...
뒤로(B)
다음(N)
취소

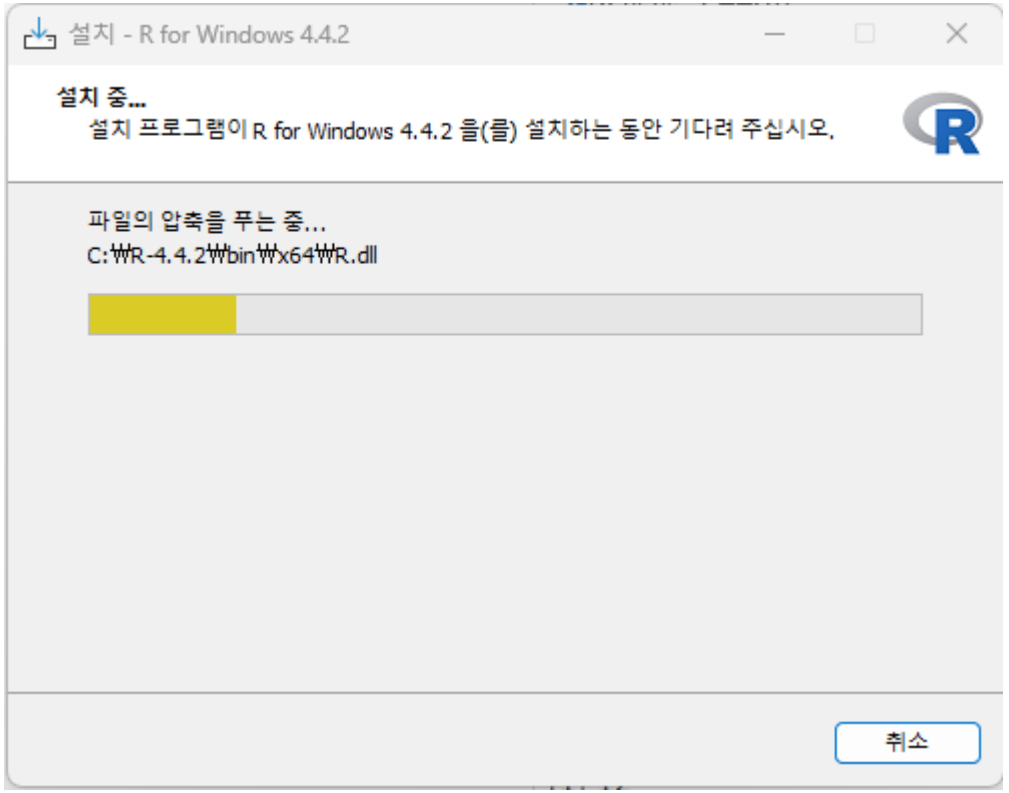

설치 - R for Windows 4.4.2
구성 요소 설치
어떤 구성 요소를 설치하시겠습니까?
설치하고 싶은 구성 요소는 선택하시고, 설치하고 싶지 않은 구성 요소는 선택을 해제하십시오. 설치를 계속할 준비가 되셨으면 "다음"을 클릭하십시오.
사용자 편의를 위한 쉬운 설치
Main Files 92.7 MB
64-bit Files 74.5 MB
Message translations 10.2 MB
선택한 구성 요소 설치에 필요한 최소 용량: 180.3 MB
뒤로(B)
다음(N)
취소

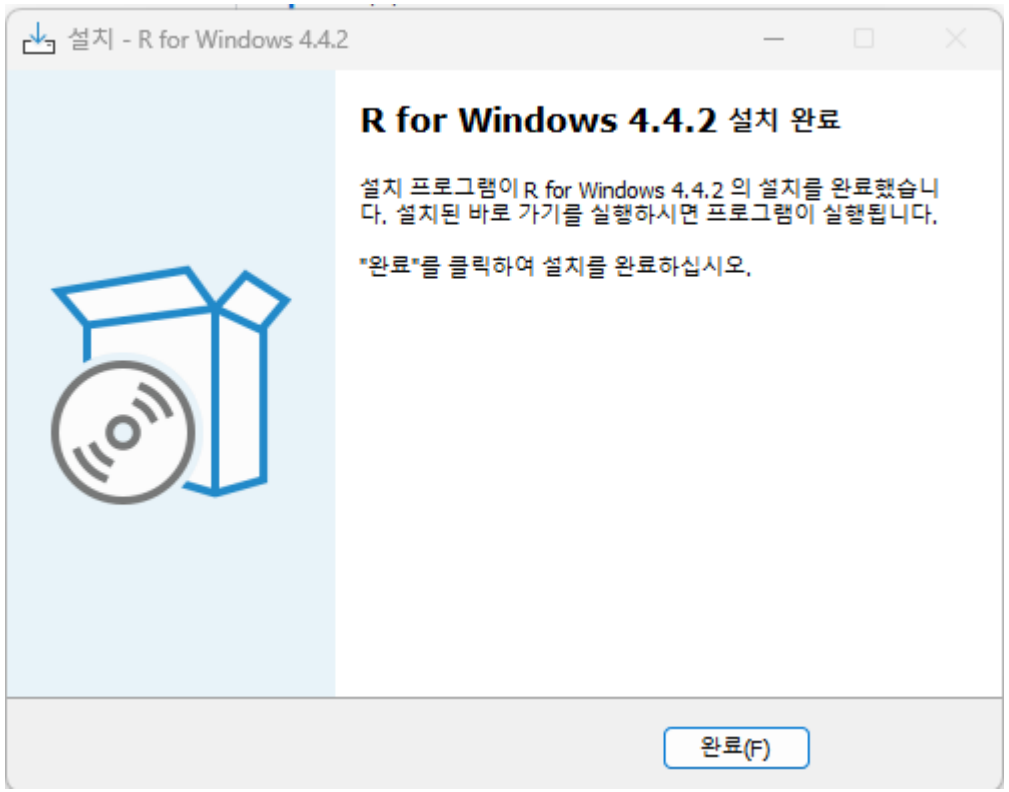
R의 설치


설치 - R for Windows 4.4.2
스타트업 옵션
스타트업 옵션을 조정하길 원하시나요?
yes 혹은 no 를 선택하신 뒤에 Next 를 눌러주세요
Yes (스타트업 조정)
No (기본값 사용)
뒤로(B)
다음(N)
취소


설치 - R for Windows 4.4.2
추가 사항 적용
어떤 사항을 추가로 적용하시겠습니까?
R for Windows 4.4.2 의 설치 과정에서 추가로 적용하고자 하는 사항을 선택하시고, "다음"을 클릭하십시오.
아이콘 생성:
바탕 화면에 아이콘 생성(D)
빠른 실행에 아이콘 생성(Q)
레지스트리 항목들:
레지스트리에 버전정보를 저장하세요
R을 .RData 파일들과 연결합니다
뒤로(B)
다음(N)
취소

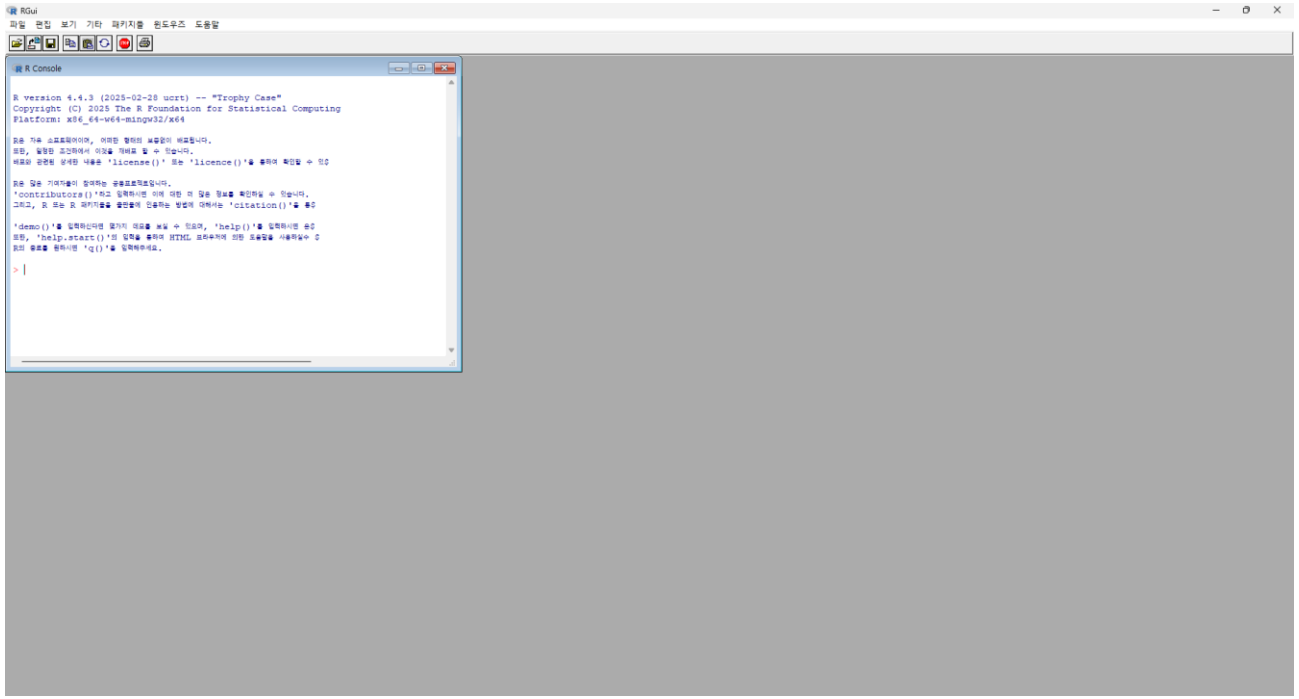
R의 설치


설치 - R for Windows 4.4.2
설치 중...
설치 프로그램이 R for Windows 4.4.2 을(를) 설치하는 동안 기다려 주십시오.
파일의 압축을 푸는 중...
C:₩R-4.4.2₩bin₩x64₩R.dll
취소

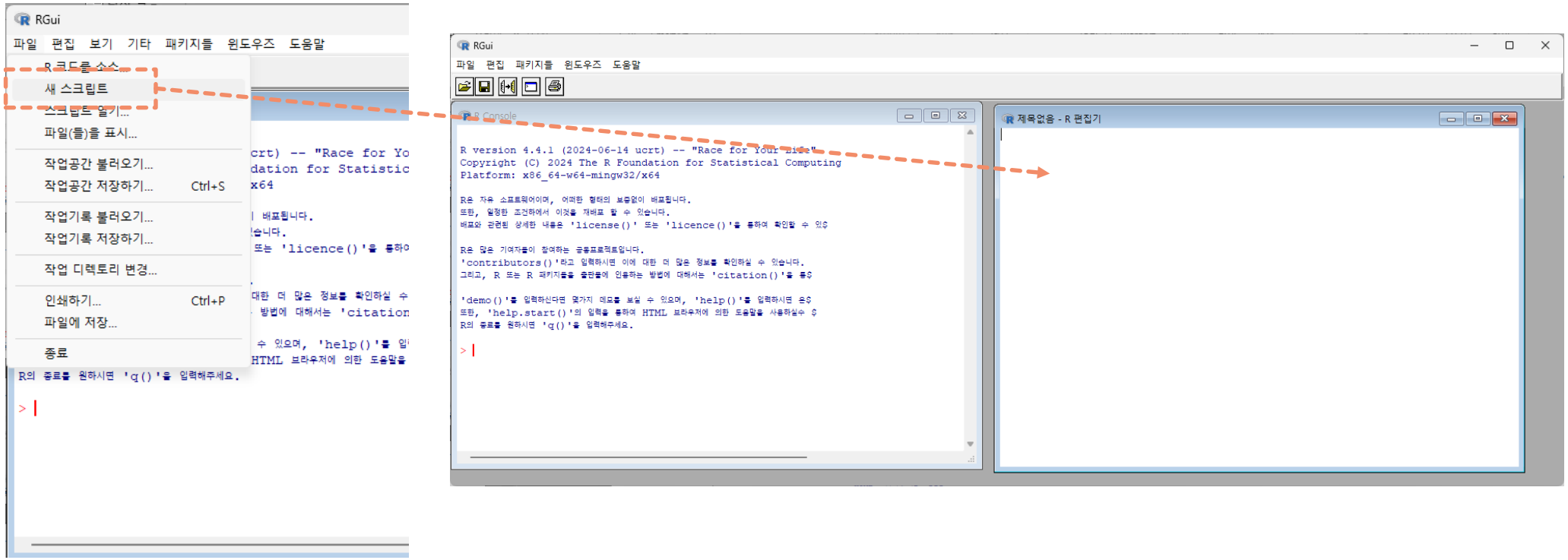

설치 - R for Windows 4.4.2
R for Windows 4.4.2 설치 완료
설치 프로그램이 R for Windows 4.4.2 의 설치를 완료했습니다. 설치된 바로 가기를 실행하시면 프로그램이 실행됩니다.
"완료"를 클릭하여 설치를 완료하십시오.
완료(F)

R의 화면 설명


R 4.4.3


RGui
파일 편집 보기 기타 패키지들 윈도우즈 도움말
R Console
R version 4.4.3 (2025-02-28 ucrt) -- "Trophy Case"
Copyright (C) 2025 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64
'demo()'를 입력하신다면 몇가지 데모를 보실 수 있으며, 'help()'를 입력하시면 온$
'help.start()'의 입력을 통하여 HTML 브라우저에 의한 도움말을 사용하실수 $
'q()'를 입력하세요.

R의 화면 설명


RGui
파일 편집 보기 기타 패키지들 윈도우즈 도움말
새 스크립트
스크립트 열기...
파일(들)을 표시...
작업공간 불러오기...
작업공간 저장하기... Ctrl+S
작업기록 불러오기...
작업기록 저장하기...
작업 디렉토리 변경...
인쇄하기... Ctrl+P
파일에 저장...
종료
R의 종료를 원하시면 'q()'을 입력해주세요.
>
RGui
파일 편집 패키지들 윈도우즈 도움말
R Console
R version 4.4.1 (2024-06-14 ucrt) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64
R은 자유 소프트웨어이며, 어떠한 형태의 보증없이 배포됩니다.
또한, 일정한 조건하에서 이것을 재배포 할 수 있습니다.
배포와 관련된 상세한 내용은 'license()' 또는 'licence()'을 통하여 확인할 수 있$
R은 많은 기여자들이 참여하는 공동프로젝트입니다.
'contributors()'라고 입력하시면 이에 대한 더 많은 정보를 확인하실 수 있습니다.
그리고, R 또는 R 패키지들을 출판물에 인용하는 방법에 대해서는 'citation()'을 통$
'demo()'를 입력하신다면 몇가지 데모를 보실 수 있으며, 'help()'를 입력하시면 온$
또한, 'help.start()'의 입력을 통하여 HTML 브라우저에 의한 도움말을 사용하실수 $
R의 종료를 원하시면 'q()'을 입력해주세요.
>
제목없음 - R 편집기

R의 문법

- R은 통계 및 데이터 분석을 위해 설계된 언어로, 벡터화 연산과 함수형 프로그래밍을 기반으로 함
- 변수와 데이터 타입
  - R에서 변수는 <- 또는 = 연산자를 사용하여 할당(=보다 <-를 권장함)

    실행할 부분에 커서를 위치하거나 여러 줄을 드래그하여 (ctrl+r)로 실행

*Rstudio는 ctrl+Enter

```
x<-10                                  #실수형 숫자, numeric
y<-as.integer(10)                      #정수형 숫자, integer
z<-FALSE                               #논리값(TRUE, FALSE), logical
w<-"Hello, World!"                     #문자열, character
f<-factor(c("one","two","three"))      #요인, factor
```

R의 문법

- R은 대소문자를 구분

```
> a<-1
> A<-2
> a
[1] 1
> A
[1] 2
```

- 숫자로 시작하는 변수명 생성 불가

```
> 1a<-2
에러: 예상하지 못한 기호(symbol)입니다. in "1a"
> a1<-2
> a1
[1] 2
```

- '#' 기호는 해당 줄을 주석 처리

```
> a<-"#"  # sharp 기호를 문자열에 넣을 경우 따옴표로 둘러줘야 한다
> a
[1] "#"
```

- 연산기호 등 기호는 변수명에 들어갈 수 없는 것들이 있음(변수명 선언 전 함수로 인식)

R의 문법

- ** str() : 개체의 구조를 보는 함수 (structure)

```
>  x
[1] 10
>  y
[1] 10
>  z
[1] FALSE
>  w
[1] "Hello, World!"
>  f
[1] one   two   three
Levels: one three two
```

```
>  str(x)
 num 10
>  str(y)
 int 10
>  str(z)
 logi FALSE
>  str(w)
 chr "Hello, World!"
>  str(f)
 Factor w/ 3 levels "one","three",..: 1 3 2
```

R의 문법

▪ 2.벡터

- R의 기본 데이터 구조는 벡터이며, 벡터 연산이 가장 빠름

```
v1<-c(1,2,3,4,5)                #숫자형 벡터
v2<-c("a","b","c")              #문자형 벡터
v3<-c(TRUE, FALSE, TRUE)        #논리형 벡터
v4<-c(1,2,"c", FALSE)           #혼합형 벡터(?)
```

R의 문법

- 혼합형 벡터는 존재하지 않으며, 하나의 벡터는 하나의 타입을 가짐

```
>  v1
[1] 1 2 3 4 5
>  v2
[1] "a" "b" "c"
>  v3
[1]  TRUE FALSE  TRUE
>  v4
[1] "1"     "2"     "c"     "FALSE"
```

```
>  str(v1)
 num [1:5] 1 2 3 4 5
>  str(v2)
 chr [1:3] "a" "b" "c"
>  str(v3)
 logi [1:3] TRUE FALSE TRUE
>  str(v4)
 chr [1:4] "1" "2" "c" "FALSE"
```

R의 문법

```
x<-c(1,2,3)
y<-c(4,5,6)

x+y                 #벡터 덧셈
x*2                 #스칼라 연산
x^2                 #스칼라 연산
sum(y)              #벡터 합계 계산
```

```
>  x+y
[1] 5 7 9
> x*2
[1] 2 4 6
> x^2
[1] 1 4 9
> sum(y)
[1] 15
```

R의 문법

▪ 3.리스트

- 벡터는 하나의 타입만 가질 수 있지만, 리스트는 여러 타입을 저장 가능

```
list2<- list(name="R", age=25,
scores=c(90,85,80))
str(list2)
list2$name
list2$age
list2$scores
```

$를 통해 리스트 내부로 접근

```
>  str(list2)
List of 3
 $ name  : chr "R"
 $ age   : num 25
 $ scores: num [1:3] 90 85 80
>  list2$name
[1] "R"
>  list2$age
[1] 25
>  list2$scores
[1] 90 85 80
```

- Python의 딕셔너리와 유사한 구조이며, 데이터프레임과 유사하지만 더 자유로움

R의 문법

▪ 4.행렬

- 행렬은 동일한 데이터 타입을 가지는 2차원 구조

```
1:9
seq(1,9)
mat2<-matrix(1:9, nrow=3)
mat3<-matrix(seq(1,9),
nrow=3, byrow=T)
mat2
mat3
```

```
>  1:9
[1] 1 2 3 4 5 6 7 8 9
>  seq(1,9)
[1] 1 2 3 4 5 6 7 8 9
>  mat2<-matrix(1:9, nrow=3)
>  mat3<-matrix(seq(1,9), nrow=3, byrow=T)
>  mat2
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>  mat3
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

R의 문법

▪ 5.데이터프레임

- 데이터 프레임은 R에서 가장 중요하며 가장 많이 쓰이는 데이터 구조

```
data2<- data.frame(name=c("A","B"),
score=c(30,40))
str(data2)
data2
```

```
>  data2<- data.frame(name=c("A","B"),
+  score=c(30,40))
>  str(data2)
'data.frame':    2 obs. of  2 variables:
 $ name : chr  "A" "B"
 $ score: num  30 40
>  data2
  name score
1    A    30
2    B    40
```

R의 문법

- 행렬을 데이터 프레임으로, 데이터프레임을 행렬로 변경할 수 있음

```
>  datmat3<-data.frame(mat3)
>  datmat3
  X1 X2 X3
1  1  2  3
2  4  5  6
3  7  8  9
>  str(datmat3)
'data.frame':   3 obs. of  3 variables:
 $ X1: int  1 4 7
 $ X2: int  2 5 8
 $ X3: int  3 6 9
```

```
>  datmat3<-as.data.frame(mat3)
>  datmat3
  V1 V2 V3
1  1  2  3
2  4  5  6
3  7  8  9
>  str(datmat3)
'data.frame':   3 obs. of  3 variables:
 $ V1: int  1 4 7
 $ V2: int  2 5 8
 $ V3: int  3 6 9
```

R의 문법

- 행렬을 데이터 프레임으로, 데이터프레임을 행렬로 변경할 수 있음

```
>  matdata2<-as.matrix(data2)
>  matdata2
     name score
[1,] "A"  "30"
[2,] "B"  "40"
>  str(matdata2)
 chr [1:2, 1:2] "A" "B" "30" "40"
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:2] "name" "score"
```

R의 문법

6. 조건문(if)

```
>   x<-4
>   if (x>=5) {                #조건 1
+   print("x는 5 이상이다")} #조건 1이 참이면 실행
>
```

```
>   x<-6
>   if (x>=5) {                #조건 1
+   print("x는 5 이상이다")} #조건 1이 참이면 실행
[1] "x는 5 이상이다"
>
```

R의 문법

▪ 6. 조건문(if else)

```
x<-4.999
if (x>=5) {                #조건 1
print("x는 5 이상이다") #조건 1이 참이면 실행
} else {
print("x는 5 미만이다") #조건 1이 거짓이면 실행
}
```

→ 같은 줄에 쓸것

```
>  x<-4.999
>  if (x>=5) {
+  print("x는 5 이상이다")
+  } else {
+  print("x는 5 미만이다")
+  }
[1] "x는 5 미만이다"
```

R의 문법

▪ 6. 조건문(if else)

```
x<-6
if (x>=5) {
print("x는 5 이상이다")}
else {
print("x는 5 미만이다")
}
```

```
>  x<-6
>  if (x>=5) {
+  print("x는 5 이상이다")}   #조건문 실행, 결과 반환
[1] "x는 5 이상이다"
>  else {                     #else 인식 오류
에러: 예기치 않은 'else'입니다 in " else"
>  print("x는 5 미만이다")    #print문 수행
[1] "x는 5 미만이다"
>  }                          #} 인식 오류
에러: 예기치 않은 '}'입니다 in " }"
>
```

** else가 중괄호의 다음 줄에 있을 경우, else를 인식하기 전에 함수가 종료되고 결과가 반환됨
else 단독으로는 함수가 아니므로, 에러가 발생함

R의 문법

▪ 6. 조건문(for)

```
>  for (i in 1:5)
+  {
+  print(i)
+  }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
>  i
[1] 5
```

```
>  k<-c(1,3,5,7,9)
>  for (i in k)
+  {
+  print(i)
+  }
[1] 1
[1] 3
[1] 5
[1] 7
[1] 9
>  i
[1] 9
```

** 사전에 정해진 sequence대로 명령을 수행

수행의 결과는 저장됨(추후 설명할 함수와의 차이)

R의 문법

6. 조건문(While)

```
>   i<-1                          #초기값 설정
>   while(i<=20)                  #while문 종료 조건
+   { if(i %% 2 ==0) {            #2로 나누었을때 나머지가 0
+     print(i)
+    }
+    i<-i+1                       #i값 1 증가
+   }
```

```
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
[1] 12
[1] 14
[1] 16
[1] 18
[1] 20
```

R의 문법

▪ 6. 조건문(While)

```
>  i<-1                  #초기값 설정
>  while(TRUE)
+  {
+   print(i)
+   i<-i+1
+   if(i>10) break   #While문 종료 조건
+  }
```

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

** 내부에 next, break 등의 명령어로 행동 설정 가능

R의 문법

6. 조건문(ifelse)

```
> x<-seq(1,10)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> ifelse(x>5,"B","A")
 [1] "A" "A" "A" "A" "A" "B" "B" "B" "B" "B"
```

** ifelse 는 for 또는 if 문과 다르게 모든 열에 대해 벡터화 하여 처리하기때문에 성능이 빠름

R의 문법

▪ 7. 사용자 정의 함수

#2로 나눈 나머지가 0인가(짝수인가)를 확인하는 조건문
!=, ==, >=, 등의 조건문은 TRUE/FALSE를 반환

```
even_number <- function(a){
 if(a %% 2 ==0) {
 return("Yes")
 } else { return("No") }
 }
```

```
>  even_number <- function(a){
+   if(a %% 2 ==0) {
+   return("Yes")
+   } else { return("No") }
+   }
>
>  even_number(2)
[1] "Yes"
>  even_number(13)
[1] "No"
```

```
>  a
에러: 객체 'a'를 찾을 수 없습니다
```

#반복문과 달리, 함수에서는 내부에서 선언된 객체명이
함수 외부로 반환되지 않음

R의 문법

- 8. 함수 저장 및 개체 저장 시 주의할 점
  - R은 함수 역시 개체 취급을 함
  - 기존의 함수 이름으로 새로운 변수를 덮어씌우는 경우 원래 함수를 사용할 수 없음
  - rm 명령어를 통해 저장된 개체를 삭제해야 기존의 함수를 다시 사용 가능함

```
>  hist
function (x, ...)
UseMethod("hist")
<bytecode: 0x000002385826b270>
<environment: namespace:graphics>
>
>  hist<-c("3","4","5")
>
>  hist
[1] "3" "4" "5"
>
> rm(list=c("hist"))
>
>  hist
function (x, ...)
UseMethod("hist")
<bytecode: 0x000002385826b270>
<environment: namespace:graphics>
```

감사합니다

Q&A

충북대학교
CHUNGBUK NATIONAL UNIVERSITY