Wk 3. Variable, Vector, Function, and Open Data

CH 3

Terminology 1 (Review)

- Accuracy vs. Precision

  - **Accuracy** is how close a measured value is to the **actual (true) value**.
    => Average ("Accuracy" in Korean?)

  - **Precision** is how close the measured values are **to each other**.
    => Variance ("Precision" in Korean?)

Calibration?

http://www.mathsisfun.com/accuracy-precision.html

Terminology 1 (2)

- Accuracy vs. Precision

eg, Thermometer:

- **You measured the temperature on testing solution (37.5 ˚C) for 10 times using your newly purchased device (100,000원):**
  - **37.2, 37.7, 37.4, 37.5, 37.9, 37.1, 37.3, 37.8, 37.6, 37.4**

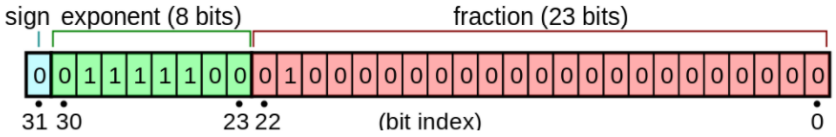
    => indeed the fraction is not meaningful! ☹

  “Hm.. I didn’t have to pay extra 90,000원”

  If you keep doing this, you need to find another job!

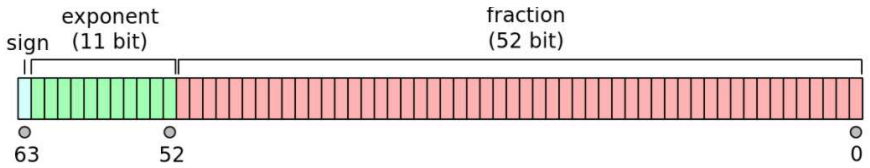
Data Precision for "CS Nerd"

- Digital Data Precision ?
  - Integer:
    - Largest integer in R: 2,147,483,647 (How may bits?)
  - Floating Point Number:
    - Single Precision FP Number: 32bits



= 0.15625 (wiki)

    - Double Precision FP Number:64bits



    - Maximum Double number in R: 1.797693e+308

"OMG. Do I have to drop this course?"

Hm.. Fraction precision follows the one of 32bits! Is it?

Data Precision in R (Integer)

- Try:
  - n1=3*1000*1000*1000 # what happens?
  - is.integer(n1)
  - is.double(n1) # auto "Type _____" occurred!

- .Machine$integer.max : 2,147,483,647
- log(.Machine$integer.max,2) : _____ bits?
- .Machine$integer.max+1L # What's the result?

Q. Is there way we can use 64 bit integer?

  - library(bit64)
  - i64=as.integer64(.Machine$integer.max)+1L # What's the result?
  - is.double(i64)
  - is.integer64(i64)

Data Precision in R (Floating Point Number)

- **c=1.234567890**

  print(c) : _____ **#**What happened?

- **print(pi) :** 3.141593

  Q.How many bits for the mantissa(fraction)? _____ bits

So indeed, R uses double precision to store a real number as default but the fraction is rounded automatically!

Q. How can we use more bits for the fraction in R?

Try print(.Machine) !

```
install.packages("Rmpfr")
library(gmp)
library(Rmpfr)
mpfr(pi,23) : 3.1415925  # basically same as print(pi)
mpfr(pi,52) : 3.1415926535897931  # the standard double precision! (Wiki)
```

Try "mpfr(pi,128)!" (If you are working for NASA)

Terminology 2

- "A **trade-off** (or **tradeoff**) is a situation that involves losing one quality or aspect of something in return for gaining another quality or aspect.

- It implies a decision to be made with full comprehension of both the upside and downside of a particular choice. And you are giving something away to get something back."

(Wiki)

Tradeoff examples

- Tradeoff between Precision and Cost-effectiveness
- In computer science, trade-offs are viewed as a tool of the trade. A program can often run faster if it uses more memory (**a space-time tradeoff**).
- Tradeoff between Performance and Security

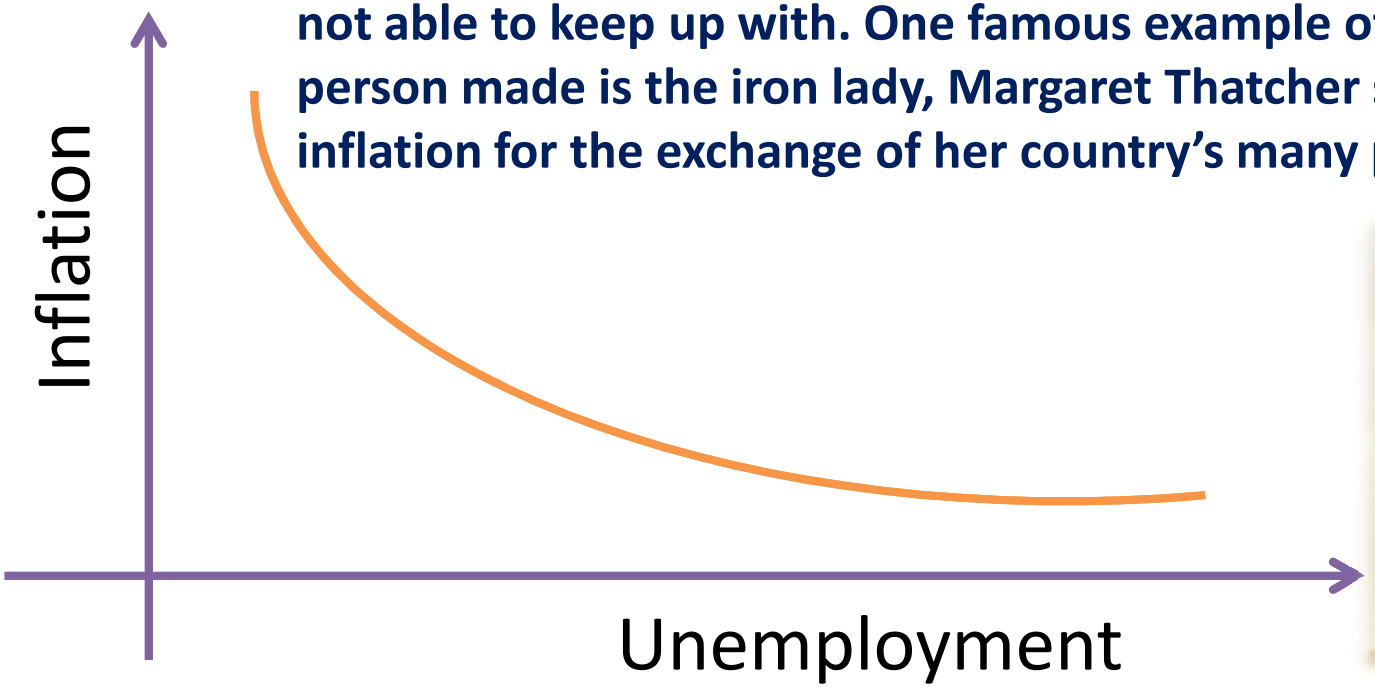
Tradeoff- example (Economics)

•"In economics the term is expressed as **Opportunity Cost**, referring to the most preferred alternative given up.

The Phillips Curve

http://www.sparknotes.com/economics/macro/measuring2/section3.rhtml

Inflation generally has a bad influence to a country, excluding the fact that it increases the GDP (Gross Domestic Product) and vacancy in employment. The sudden increase in the price of dairy products is something that the most of the population is not able to keep up with. One famous example of the tradeoff a person made is the iron lady, Margaret Thatcher stopping the inflation for the exchange of her country's many people's jobs.



“R” Time:

Programming 101

IF-ELSE EXAMPLE: R (PROGRAMMING 101)

Q. Have you now fully understood the Truth Table?

```
num <- floor(runif(1,1,100))
Choice <- readline(prompt="choose: odd / even:")
cat("num:",num," choice:",choice)

mod_num <- floor(num%%2)
if(mod_num==1 && choice=='odd' || mod_num==0 && choice=='even') {
  print("Correct :)")
} else print("Sorry :(")
```

Q. Operator Precedence?

Q. Operator Associativity?

IF-ELSE EXAMPLE: JAVA

Boolean Algebra (Truth Table) is the fundamental of Programming!

Q. How about this: "if(!p || (q && q<3))"

i.e., Online Course Registration

```
function boolean register(Course course, Student std){
  /* register a student who has not taken the course or who took it but got less than B grade */
        if(!std.did_I_take_it(course.id) || (course.get_grade(std.id,course.id) && course.get_grade(std.id,course.id)<3)) {
              register(course.id,std.id);
              return true;
        } else return false;
}
```

BECOME A WRITER WHO WRITES BEAUTIFUL CODES!


```
if(c==1){
    if(u==3) {
        print("User Won")
    } else if(u==1) {
        print("Draw")
    } else if(u==2) {
        print("Computer won")
   }
}
```

```
R <- 1 # Use a constant! It's there for you ☺
S <- 2
P <- 3
if(c==R){ # Com: Rock
    if(u==P) {  #User: Paper
        print("User Won ☺")
    } else if(u==R) { #User: Rock
        print("Draw")
    } else {  #User: Scissor
        print("Computer won ☹")
   }
}
```

01

Variable

VARIABLE


```
c <- a + b
```

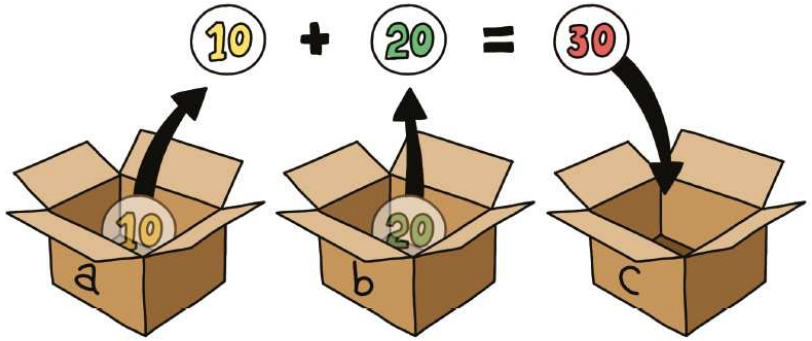


그림 3-4 c <- a + b의 실행 과정

Q, What about Constant?

- It's a container!
- You can store a number, a character, a string, a Boolean value, or a special value like _____

Rules for a Variable Name

- **It should start with "." or an alphabet:**

  eg) avg, .total : **okay!**

  eg) 12th : okay?

- **After that, you can use ".", "_", and numbers:**

  eg) value.1, sub_total, d10 : **okay!**

  eg) this-data, this@data : okay?

- **R is a "Case-Sensitive" Language!**

  eg) var_A $\neq$ var_a !

- **No space inside the name:**

  eg) first ds : NOT okay!

Name with good names!

- **Readability is important! (Let others/yourself understand it)**

```
a <- 850      # what?
math.sum <- 850  # aha! storing math score sum!
```

- **Can't use Keywords! Avoid the "built-in" names.**

```
for=3 # keyword!
pi <- 3.14       # Oh where is my pie?
sqrt <- 340      # sqrt() is the name of a built-in function
```

Try some operations on variables

> 1/0 : Inf # In other languages, what kind of error is this? _____
> -2/0: -Inf
> sqrt(-5) : [1] NaN Warning message: In sqrt(-5) : NaNs produced
> a=1
> b='2'
> a+b : What happens?

> c=1.2
> a+b

02

Vector

VECTOR

- **It is same as "Array" in other programming languages, which stores the same type of data consecutively supporting "random access"(?) of elements by indices.**

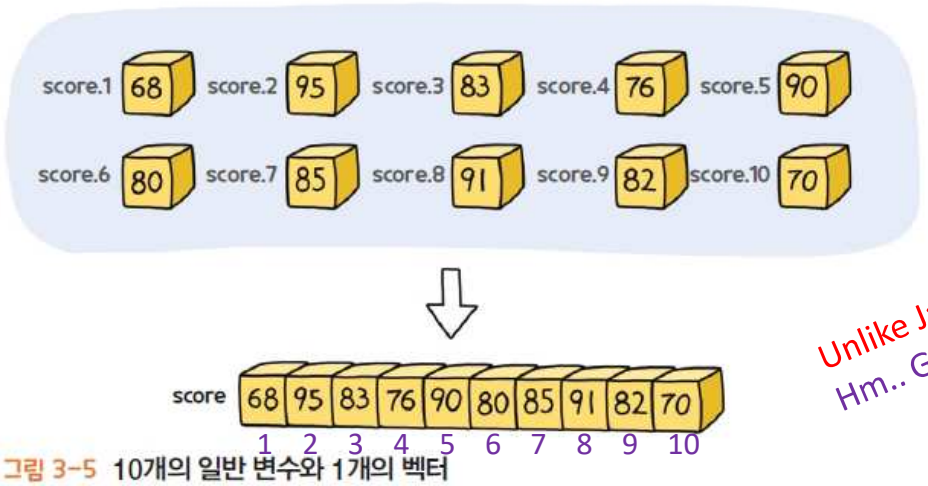


그림 3-5 10개의 일반 변수와 1개의 벡터

Unlike Java and others, the index starts w/ **1**!
Hm.. Good. But why?

Try some operations on vectors


```
x <- c(1,2,3)                    # numeric vector
y <- c('a','b','c')              # character vector
z <- c(TRUE,TRUE,FALSE,TRUE)     # Logical(Boolean) vector
x                                # all elements
X[0]                             # ?
Y[2]                             # 2nd element
z[5]                             # ?
```

Operations on vectors


```
x <- c(1,2,3)                    # numeric vector
y <- c('a','b','c')              # character vector
z <- c(TRUE,TRUE,FALSE,TRUE) # logical(Boolean) vector
x                                # all elements
X[0]                             # ?
Y[2]                             # 2nd element
z[5]                             # ?

w <- c(1,2,3, 'a','b','c') # All elements should be same type !
```

Sequences


```
> v1 <- 50:90
> v1
[1] 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
[23] 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
```

```
> v2 <- c(1,2,5, 50:90)
> v2
[1]  1  2  5 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
[23] 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
```

Sequences (1)


```
> v1 <- 50:90
> v1
[1] 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
[23] 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
```

```
> v2 <- c(1,2,5, 50:90)
> v2
[1]  1  2  5 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
[23] 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
```

Sequences (2)


```
> v3 <- seq(1,101,3)          # first, last, interval
> v3
[1]  1  4  7 10 13 16 19 22 25 28 31 34 37 40 43 46
[17] 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94
[33] 97 100
>
> v4 <- seq(0.1,1.0,0.1)      # first, last, interval
> v4
 [1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Repetition using “times”


```
> v5 <- rep(1,times=5)        # 5 1s
> v5
[1] 1 1 1 1 1
>
> v6 <- rep(1:5,times=3)      # 1, 2, 3, 4, 5 X 3 times
> v6
 [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
>
> v7 <- rep(c(1,5,9), times=3)       # 1, 5, 9 x 3 times
> v7
[1] 1 5 9 1 5 9 1 5 9
```

Repetition using “each”


```
> v8 <- rep(c('a','b','c'), each = 3)
> v8
[1] "a" "a" "a" "b" "b" "b" "c" "c" "c"
```

Named Vector


```
> absent <- c(8,2,0,4,1) # Number of absences
> absent
> names(absent)                    # Names of Vector
> names(absent) <- c('Mon','Tue','Wed','Thu','Fri') # give names

> absent
```

Named Vector: access elements by name


```
> sales <- c(640,720,680,540)          # Jan~Apr Income
> names(sales) <- c('M1','M2','M3','M4') # Give names
> sales                                # 1~4
> sales[1]                             # Jan
> sales[2:4]                           # Feb~Apr
> sales['M2']                           # Feb
> sales[c('M1','M4')]                   # Jan, Apr
> v1 <- c(1,5,7,8,9)
> v1[c(1,5)] <- c(10,20)   # replace only 1st and 5th values
```

LAB #1. CD Deposit Yield Calculator (1)

Name	kim	lee	park	choi	seo
Deposit	5,000,000	4,500,000	4,000,000	5,500,000	6,000,000
Interest Rate	3.5%	3%	4%	5%	4.5%
Period	2	2	5	7	4

Total Amount= Deposit × $(1+ \text{Interest Rate}/100)^{\text{period}}$

LAB #1. CD Deposit Yield Calculator (2)

1. **Vector generation for customers, deposits, rates, and periods.**

```
customer <- c('kim', 'lee', 'park', 'choi', 'seo')
deposit <- c(5000000, 4500000, 4000000, 5500000, 6000000)
rate <- c(3.5, 3, 4, 5, 4.5)
period <- c(2, 2, 5, 7, 4)
```

2. **Apply names**

```
names(deposit) <- customer
names(rate) <- customer
names(period) <- customer
```

3. **Define a variable called “who”**

```
who <- 'kim'
```

LAB #1. CD Deposit Yield Calculator (3)

4. Formula

```
sum <- deposit[who] * ( 1 + rate[who] / 100)^ period[who]
sum
```

```
    kim
5356125
```

5. Try calculating everybody's result using a loop:

Today's Proverb (Review)

- "Give me six hours to chop down a tree and I will use the first four sharpening the axe"

(Abraham Lincoln)

Gmshtcdat alwstffsta

https://wise4edu.com/init/#s4

Today's Proverb

- "A rolling stone gathers no moss"

Arsgnm

Art by Nicola Temple

https://wise4edu.com/init/#s24

03

Function

Function

A Function can be considered as a little Computer: I → P → O

```
f(x) = 2x + 1
```

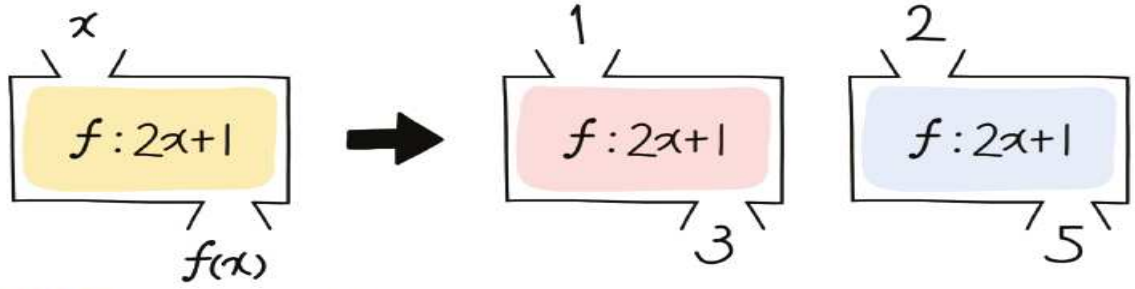


그림 3-8 함수 f(x)=2x+1의 실행

Functions are basic building blocks for “D & C”!

Q.What’s “D&C”?

Function: examples


```
factorial(5)
abs(-3)
log(16,2)
cowsay::say("function is fun","random")

scores <- c(80,95,74,83,60,98,81,77,79,94)
mean(scores)
sqrt(var(scores)) # what is it?
sort(scores,TRUE)
hist(scores)
```

paste()

```
y <- 2021
m <- 1:12
yr_mt <- paste(y,m, sep='/')
```

```
[1] "2021/1"  "2021/2"  "2021/3"  "2021/4"  "2021/5"  "2021/6"
"2021/7"  "2021/8"  "2021/9"  "2021/10"
[11] "2021/11" "2021/12"
```

Defining a function


```
scores <- c(80,95,74,83,60,98,81,77,79,94)
sum(scores)

my_sum <- function(data){
  result=0 # initialization
  for(i in 1:length(data)){
    result=result+data[i]
  }
  return(result)
}

print(my_sum(scores))
```

LAB #2. Make my_mean()

Open Data

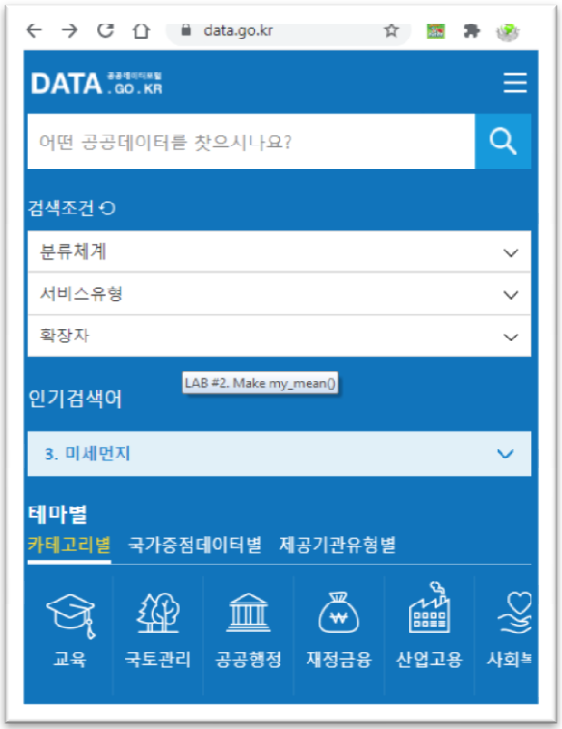
Formats: xml, json, csv

Public Big Data

Open Data #1 – data.go.kr

- Let's try some keywords
  like "코로나"

Q. What are the data formats?



Open Data #1 – data.go.kr

- 인천광역시_소비 데이터 (CSV):
  https://www.data.go.kr/data/15076578/fileData.do

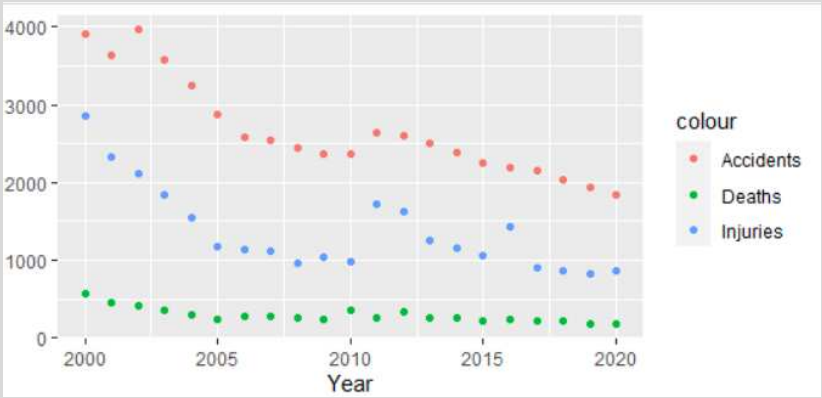
- 한국도로공사_교통사고통계 (CSV):
  https://www.data.go.kr/data/15045638/fileData.do

- 시도별 백신 접종율 (XML):
  https://nip.kdca.go.kr/irgd/cov19stats.do?list=sido

한국도로공사_교통사고통계 (CSV)

연도	사고	사망	부상
2000	3910	569	2845
2001	3638	456	2331
2002	3957	421	2115
2003	3585	348	1843
2004	3242	300	1555
2005	2880	249	1170
2006	2583	284	1131
2007	2550	283	1114
2008	2449	265	955
2009	2374	248	1031
2010	2368	353	983
2011	2640	265	1731
2012	2600	343	1619
2013	2496	264	1253
2014	2395	253	1148
2015	2251	223	1054
2016	2195	239	1424
2017	2145	214	911
2018	2030	227	858
2019	1931	176	830
2020	1834	179	861

```
library(ggplot2)
tdata <-
read.csv("D:/한국도로공사_교통사고통계_20201231.csv")
class(tdata)
View(tdata)

ggplot(data=tdata, aes(Year)) +
  geom_point(aes(x = Year, y=Accidents,colour = "Accidents")) +
  geom_point(aes(x = Year, y=Deaths,colour = "Deaths"))+
  geom_point(aes(x = Year, y=Injuries,colour = "Injuries"))
```



Let's try one!

- 경기도_정류장 주변도로 미세먼지빅데이터 기반 대응시스템 공공데이터 (JSON):

  https://www.data.go.kr/data/15062769/openapi.do

Let's try one!

샘플코드

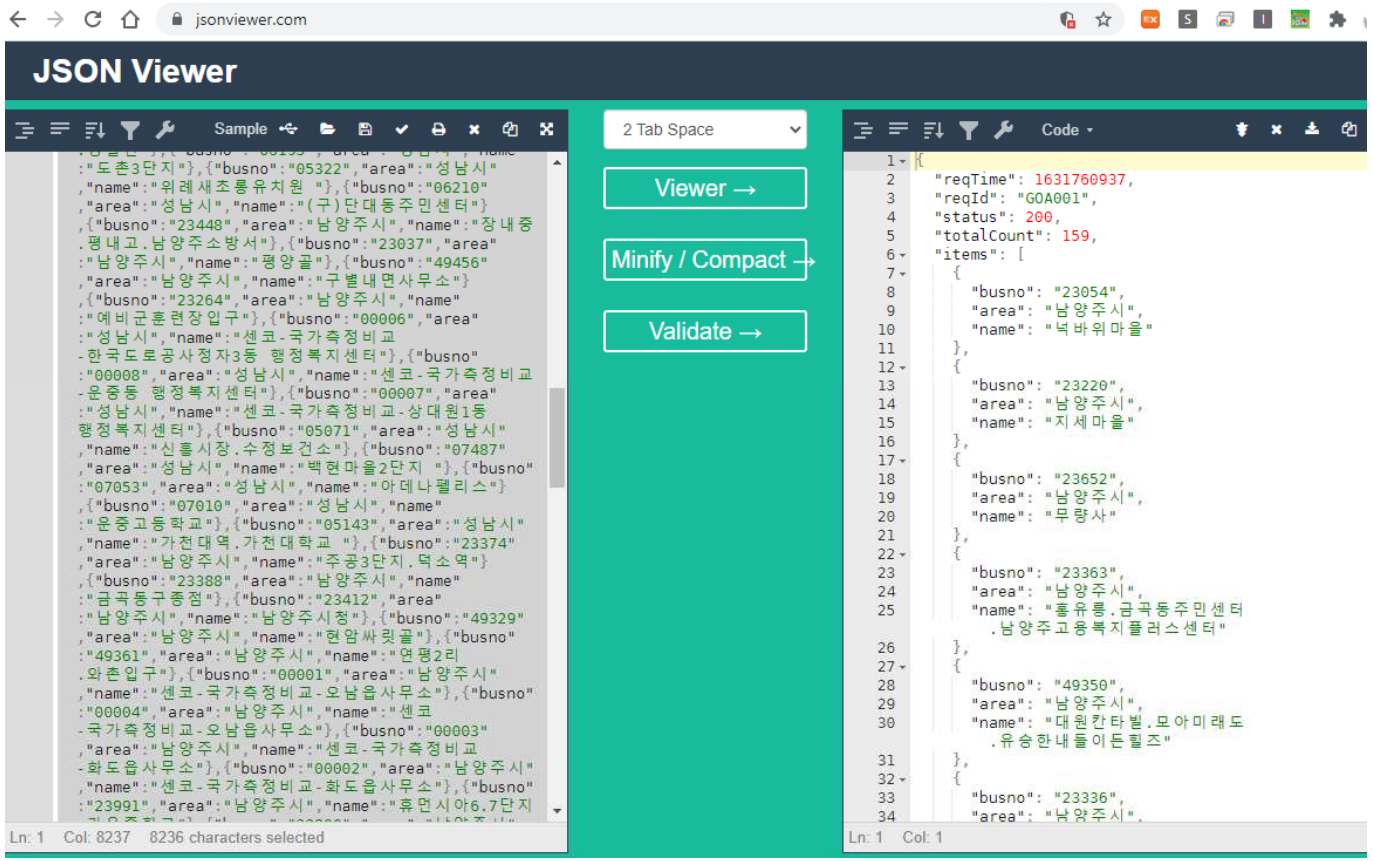
Java | Javascript | C# | PHP | Curl | Objective-C | Python | Nodejs | R

```
/* R 샘플 코드 */

install.packages("httr") # HTTP통신을 위한 패키지 설치

library(httr)
GET('http://apis.data.go.kr/6410000/GOA/GOA001?ServiceKey=서비스키&type=json&numOfRows=10&pageNo=1')
```

```
install.packages("httr") # HTTP통신을 위한 패키지 설치
library(httr)
serviceKey="**********************************************"
resp<-
    GET(paste('http://apis.data.go.kr/6410000/GOA/GOA001?ServiceKey=',ser
    viceKey,'&type=json&numOfRows=10&pageNo=1',sep=''))
jsonRespParsed<-content(resp,as="parsed")
View(jsonRespParsed$items)
```

jsonRespParsed$items	list [159]	List of length 159
[[1]]	list [3]	List of length 3
busno	character [1]	'23054'
area	character [1]	'남양주시'
name	character [1]	'넉바위마을'
[[2]]	list [3]	List of length 3

JSON Viewer


jsonviewer.com
JSON Viewer
Sample
2 Tab Space
Viewer →
Minify / Compact →
Validate →
Code
Ln: 1 Col: 8237 8236 characters selected
Ln: 1 Col: 1

HW #1 due by 9/23

1. Think a/b 3 Trade-offs:
   A. One interesting tradeoff in your daily life
   B. One interesting tradeoff of your major
   C. One interesting tradeoff of our society
   (don't just do googling !)
2. Average and Standard Deviation over 100M test scores:
   A. Generate 100,000,000 test scores with $\mu(80,15)$ and save into "scores" (vector).
   B. What is the size of the data in MB?
   C. Use my_sum() to print the sum. Is it working? What is the data type of the sum?
   D. Make my_avg() and my_sd() and see if it is close to $\mu(80,15)$.
   What do you call this type of distribution?
   E. Include the **Histogram**.
3. Get a public data Set of your interest from https://www.data.go.kr/
   Get a permission! And include the "**meta data**" like the screenshot.

*** Submit a MS Word document. Write everything in English. Include the references if there is any.**

Meta data Example (HW #1-3)


```
{
  "name": "경기도_정류장 주변도로 미세먼지빅데이터
    기반 대응시스템 공공데이터 조회",
  "description": "정류장 주변도로 미세먼지빅데이터
    기반 대응시스템 공공데이터 조회",
  "url": "https://www.data.go.kr/data/15062769/openapi
    .do",
  "keywords": [
    "빅데이터,버스정류장,미세먼지"
  ],
  "license": "https://data.go.kr/ugs
    /selectPortalPolicyView.do",
  "creator": {
    "name": "경기도",
    "contactPoint": {
      "contactType": "정보기획담당관",
      "telephone": "+82-03180082821",
      "@type": "ContactPoint"
    },
    "@type": "Organization"
  },
  "distribution": [
    {
      "encodingFormat": "JSON",
      "contentUrl": "https://www.data.go.kr/data
        /15062769/openapi.do",
      "@type": "DataDownload"
    }
  ],
  "@context": "https://schema.org",
  "@type": "Dataset"
}
```

Next Week Topics

- Control / Loop statements
  - We've already learned this ☺
    So let's practice w/ Big data!
- User Defined Function
  - **Call by Value vs. Call by Reference**
  - **Recursive Function in R**
  - **and more**

To-dos for Next Class (Flipped Learning)

- **Try enough R codes!**

  like a "Poet"!

- **Textbook**

  – Read ch7