R 원칙

- R 상에 존재하는 모든 것은 오브젝트 object이다.
- R 상의 모든 실행은 함수 fuction에 의해서만 이루어진다.

최소 클래스 atomic class

- 문자 character : c("한남대", "꿈")
- 숫자 numeric (real) : c(1, 3.5, -0.1)
- 정수 integer : c(2)
- 논리 logical : TRUE, FALSE
- 복수 complex : c(2+3i) 주의, 3*i 차이 - 3과 오브젝트 i를 곱한다.

```
a<-c(3+2i); print(a); typeof(a); class(a); length(a); attributes(a)
b<-c(-2, 0.1, 3); print(b); typeof(b); class(b); length(b); attributes(b)
c<-c(1:5) ; print(c); typeof(c); class(c); length(c); attributes(c)
d<-(3>5); print(d); typeof(d);class(d);length(d); attributes(d)
e<-c("who", "what", 39); print(e); typeof(e);class(e);length(e); attributes(e)
```

```
> a<-c(3+2i);
[1] 3+2i
[1] "complex"
[1] "complex"
[1] 1
NULL
```

```
> b<-c(-2, 0.1, 3);
[1] -2.0  0.1  3.0
[1] "double"
[1] "numeric"
[1] 3
NULL
```

```
> c<-c(1:5) ; p
[1] 1 2 3 4 5
[1] "integer"
[1] "integer"
[1] 5
NULL
```

```
> d<-(3>5); print(d); t
[1] FALSE
[1] "logical"
[1] "logical"
[1] 1
NULL
> e<-c("who", "what", 3
es(e)
[1] "who"  "what" "39"
[1] "character"
[1] "character"
[1] 3
NULL
```

- typeof() : 클래스 타입, double= 실수
- class() : 클래스 타입이나 typeof() 보다 큰 개념
- attributes() : 클래스 속성 (메타정보 포함), 메타정보(2개 이상의 최소 클래스가 이루어지고 행과 열의 형식을 가진 데이터) 포함하지 않은 클래스는 NULL
- length() : 클래스 길이, 열의 길이를 의미함
- dim() : 데이터 프레임 행과 열의 차원, 데이터 프레임이 아닌 경우 NULL 출력

현재 사용 중인 오브젝트 리스트를 알고 싶다면...

```
> ls()
[1] "a" "b" "c" "d" "e"
```

```
data()  #내장 데이터 출력
typeof(AirPassengers); class(AirPassengers); attributes(AirPassengers)
length(AirPassengers); dim(AirPassengers)
ds<-read.csv("http://wolfpack.hnu.ac.kr/Stat_Notes/example_data/SMSA_USA.csv")
typeof(ds); class(ds); attributes(ds)
length(ds); dim(ds)
```

```
> typeof(AirPassengers)
[1] "double"
> class(AirPassengers)
[1] "ts"
> attributes(AirPassengers)
$`tsp`
[1] 1949.000 1960.917   12.000

$class
[1] "ts"

> length(AirPassengers); dim(AirPassengers)
[1] 144
NULL
```

- ts : times series 시계열 데이터, 데이토 프레임이 아니므로 차원 dim NULL 출력
- 데이터 프레임 data frame 은 list 임

```
> typeof(ds)
[1] "list"
> class(ds)
[1] "data.frame"
> attributes(ds)
$`names`
 [1] "city_name"          "jan_te
 [4] "humidity"           "rainfa
 [7] "education"          "pop_de
[10] "white_color_ratio"  "popula
[13] "household_income"   "HCPot"
[16] "SO2Pot"             "southe

$class
[1] "data.frame"

$row.names
 [1]  1  2  3  4  5  6  7  8  9
[22] 22 23 24 25 26 27 28 29 30
[43] 43 44 45 46 47 48 49 50 51

> length(ds); dim(ds)
[1] 17
[1] 59 17
```

데이터 구조 data structure

- (atomic) vactor 벡터 : 문자, 숫자, 논리의 클래스 집합

```
x1<-vector("numeric", length=9);x1
x2<-vector("character", length=3);x2
```

```
> x1<-vector("numeric", length=9);x1
[1] 0 0 0 0 0 0 0 0 0
> x2<-vector("character", length=3);x2
[1] "" "" ""
```

vector() 함수에 의해 빈 벡터 오브젝트를 정의할 수 있음

벡터 입력(정의)는 c() 함수를 이용하는 것이 편리하다. (권고)

```
y <- c(3:9); y
y1 <- c("hong", "nam"); y1; typeof(y1)
y2 <- c(1+6, 2+(4+4i),"char") ; y2; typeof(y2)
```

✴ 문자와 숫자 혼합인 경우 "문자"로 인식함

✴ 벡터 데이터 위치는 오브젝트 이름[위치번호]로 인지한다.

```
> y <- c(3:9); y
[1] 3 4 5 6 7 8 9
> y1 <- c("hong", "nam"); y1; typeof(y1)
[1] "hong" "nam"
[1] "character"
> y2 <- c(1+6, 2+(4+4i),"char") ; y2; typeof(y2)
[1] "7"    "6+4i" "char"
[1] "character"
```

```
> y[3];y1[2]
[1] 5
[1] "nam"
```

✴ matrix : 행과 열 형식의 숫자 데이터

```
m <- matrix(seq(2,12,2),nrow=2,ncol=3,byrow=T); m
typeof(m);class(m); dim(m);length(m)
```

```
> m <- matrix(seq(2,12,2),nrow=2,ncol=3,byrow=T); m
     [,1] [,2] [,3]
[1,]    2    4    6
[2,]    8   10   12
> typeof(m);class(m); dim(m);length(m)
[1] "double"
[1] "matrix"
[1] 2 3
[1] 6
```

byrow는 데이터 입력 순서를 행부터 시작하는 옵션 설정임

✴ list : 클래스 2개 이상의 집합, 리스트 순환 recursive 리스트라고 도 함

```
x <- list(1, "a", TRUE, (3<2), 1 + (0+4i)); x
typeof(x); class(x); x[4]
```

```
> x <- list(1, "a", TRUE, (3<2), 1 + (0+4i)); x
[[1]]
[1] 1

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

```
> typeof(x); class(x); x[4]
[1] "list"
[1] "list"
[[1]]
[1] FALSE
```

✳ data frame : 변수명과 행렬 형식의 데이터, 통계분석 함수 사용 시 대부분 데이터 프레임 구조를 가지고 있어야 사용 가능하다. 적절한 분석 함수를 사용 했음에도 불구하고 결과 출력이 되지 않는 경우는

```
data()
head(AirPassengers,3)
typeof(AirPassengers); class(AirPassengers);
is.data.frame(AirPassengers); names(AirPassengers); dim(AirPassengers)
head(ChickWeight,3)
typeof(ChickWeight); class(ChickWeight);
is.data.frame(ChickWeight); names(ChickWeight); dim(ChickWeight)
```

```
> head(AirPassengers,3)
[1] 112 118 132
> typeof(AirPassengers); class(AirPassengers);
[1] "double"
[1] "ts"
> is.data.frame(AirPassengers); names(AirPasser
[1] FALSE
NULL
NULL
```

ts(time series) 클래스는 데이터 프레임이 아님, 그래서 names() - 데이터 프레임 내 변수이름 출력, 데이터 길이 length(), 데이터 행과 열의 차원 dim() 결과 없음

데이터 프레임 구조는 typeof() = list, class()는 "data.frame"으로 출력되고 차원 dim() 출력된다.

is.data.frame() 함수는 논리함수로 데이터 프레임 오브젝트는 TRUE 그렇지 않으면 FALSE 라고 출력된다

```
> head(ChickWeight,3)
  weight Time Chick Diet
1     42    0     1    1
2     51    2     1    1
3     59    4     1    1
> typeof(ChickWeight); class(ChickWeight);
[1] "list"
[1] "nfnGroupedData" "nfGroupedData"  "grou
> is.data.frame(ChickWeight); names(ChickW
[1] TRUE
[1] "weight" "Time"   "Chick"  "Diet"
[1] 578   4
```

✳ factor : 범주형(분류형, 예 거주지역, 직업군) 변수에 관련된 함수, 통계 분석 함수 lm(), glm()에 필요한 오브젝트임

```
is.factor(ChickWeight$Diet)
table(ChickWeight$Diet);
is.factor(ChickWeight$Time)
table(ChickWeight$Diet)
z<-factor(ChickWeight$Diet); is.factor(z)
```

```
> is.factor(ChickWeight$Diet)
[1] TRUE
> table(ChickWeight$Diet);

  1   2   3   4
220 120 120 118
> is.factor(ChickWeight$Time)
[1] FALSE
> table(ChickWeight$Time)

 0  2  4  6  8 10 12 14 16 18 20 21
50 50 49 49 49 49 49 48 47 47 46 45
> z<-factor(ChickWeight$Time); is.factor(z)
[1] TRUE
```

```
lm(data=ChickWeight, weight~Diet)
lm(data=ChickWeight, weight~Time)
```

Diet 군에 따른 닭의 몸무게 차이 분산분석을 위하여 lm() 함수 사용시 Diet 변수가 factor이므로 문제가 없음

그러나 Time은 factor가 아니므로 분산분석이 불가능함-> 그러나 실제 결과는 나오는데 이는 Time이 numeric(숫자)으로 입력되어 있어 회귀분석을 한 결과가 출력된다. 매우 조심할 필요가 있음->

```
> summary(lm(data=ChickWeight, weight~Time))

Call:
lm(formula = weight ~ Time, data = ChickWeight)

Residuals:
     Min       1Q   Median       3Q      Max
-138.331  -14.536    0.926   13.533  160.669

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  27.4674     3.0365   9.046   <2e-16 *
Time          8.8030     0.2397  36.725   <2e-16 *
```

<— 회귀분석 결과

분산분석 결과 —>

```
> summary(lm(data=ChickWeight, weight~factor(Time)))

Call:
lm(formula = weight ~ factor(Time), data = ChickWeight)

Residuals:
    Min      1Q  Median      3Q     Max
-144.69  -11.93   -0.06   11.76  154.31

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       41.060      5.472   7.503 2.43e-13 ***
factor(Time)2      8.160      7.739   1.054   0.2922
factor(Time)4     18.899      7.779   2.430   0.0154 *
factor(Time)6     33.246      7.779   4.274 2.25e-05 ***
```

✴ table : 데이터의 빈도분석 결과를 출력한다.

```
names(mtcars)
table(mtcars$cyl)
prop.table(table(mtcars$cyl))
table(mtcars$cyl,mtcars$gear)
margin.table(table(mtcars$cyl,mtcars$gear),1) # 2=column marginal
prop.table(table(mtcars$cyl,mtcars$gear))
prop.table(table(mtcars$cyl,mtcars$gear),1) # 2=column percentage
```

```
> names(mtcars)
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"
> table(mtcars$cyl)

 4  6  8
11  7 14
> prop.table(table(mtcars$cyl))

      4       6       8
0.34375 0.21875 0.43750
> table(mtcars$cyl,mtcars$gear)

     3  4  5
  4  1  8  2
  6  2  4  1
  8 12  0  2
> margin.table(table(mtcars$cyl,mtcars$gear),1) # 2=column marginal

 4  6  8
11  7 14
> prop.table(table(mtcars$cyl,mtcars$gear))

          3       4       5
  4 0.03125 0.25000 0.06250
  6 0.06250 0.12500 0.03125
  8 0.37500 0.00000 0.06250
> prop.table(table(mtcars$cyl,mtcars$gear),1) # 2=column percentage

             3          4          5
  4 0.09090909 0.72727273 0.18181818
  6 0.28571429 0.57142857 0.14285714
  8 0.85714286 0.00000000 0.14285714
```