Homework assignment #1

2017100057 / 이영노

September 22, 2022

2.8.

(7) We can create vector using `seq()`.

```
ans7=seq(1,100,2)
ans7
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
## [26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

(8) In function `seq()`, we can control the amount of increments by following code. Length of vector is "86".

```
ans8=seq(6,55,4/7)
head(ans8)
```

```
## [1] 6.000000 6.571429 7.142857 7.714286 8.285714 8.857143
```

```
length(ans8)
```

```
## [1] 86
```

(9) Class of the following object `a <- seq(1,10,0.5)` is "numeric".

```
a=seq(1,10,0.5)
class(a)
```

```
## [1] "numeric"
```

(10) Class of the following object `a <- seq(1,10)` is "integer".

```
a=seq(1,10)
class(a)
```

```
## [1] "integer"
```

(11) Confirmed by following code below.

```
class(1L)
```

```
## [1] "integer"
```

(12) Originally vector x was character vector, but we can coerce it into integer vector by using code `as.numeric()`.

```
x<-c("1","3","5")
x=as.integer(x)
x
```

```
## [1] 1 3 5
```

we can confirm whether vector `x` is integer or not by `class()` function. Class is "integer"

```
class(x)
```

```
## [1] "integer"
```

2.10.

(5) Function `rank()` results out the rank of each elements. (Default is ascending order.)

Function `data.frame()` results out the dataframe within its column inputs as its arguments.

```
library(dslabs)
```

```
## Warning: 패키지 'dslabs'는 R 버전 4.1.3에서 작성되었습니다
```

```
data("murders")

ranks=rank(murders$population)
my_df=data.frame(name=murders$state, rank=ranks)
head(my_df)
```

```
##          name rank
## 1     Alabama   29
## 2      Alaska    5
## 3     Arizona   36
## 4    Arkansas   20
## 5  California   51
## 6    Colorado   30
```

(6) To make state orders from least populous to most populous, we first have to make list of orders `ind`, then filter it by assigning its orders to the rows.`my_df=my_df[ind,]`

```
ind=order(murders$population)
my_df=my_df[ind,]
head(my_df)
```

```
##                   name rank
## 51             Wyoming    1
## 9  District of Columbia    2
## 46             Vermont    3
## 35        North Dakota    4
## 2               Alaska    5
## 42        South Dakota    6
```

(7) Assign logical vector telling whether there is NA or not by `ind=is.na(na_example)`. Then sum the total `True` value by `sum(ind)`.

Total number of `NA`s of `na_example` is "145" proven by code below.

```
data("na_example")
str(na_example)
```

```
##  int [1:1000] 2 1 3 2 1 3 1 4 3 2 ...
```

```
ind=is.na(na_example);

sum(ind) # sum of NA in `ind`
```

```
## [1] 145
```

(8) Average except NA is 2.301754 by code below.

```
library(dplyr)
```

```
## Warning: 패키지 'dplyr'는 R 버전 4.1.3에서 작성되었습니다
```

```
##
## 다음의 패키지를 부착합니다: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
na=is.na(na_example) # NA is True
mean(na_example[!is.na(na_example)]) # non-NA is True. Filtering by logical operations.
```

```
## [1] 2.301754
```

2.14.

(1) Computed per 100,000 murder rate for each state, stored it into `murder_rate`, created logical vector `low`.

```
library(dslabs)
data(murders)

murder_rate=murders$total/murders$population*100000
print(murder_rate)
```

```
##  [1]  2.8244238  2.6751860  3.6295273  3.1893901  3.3741383  1.2924531
##  [7]  2.7139722  4.2319369 16.4527532  3.3980688  3.7903226  0.5145920
## [13]  0.7655102  2.8369608  2.1900730  0.6893484  2.2081106  2.6732010
## [19]  7.7425810  0.8280881  5.0748655  1.8021791  4.1786225  0.9992600
```

```
## [25]  4.0440846  5.3598917  1.2128379  1.7521372  3.1104763  0.3798036
## [31]  2.7980319  3.2537239  2.6679599  2.9993237  0.5947151  2.6871225
## [37]  2.9589340  0.9396843  3.5977513  1.5200933  4.4753235  0.9825837
## [43]  3.4509357  3.2013603  0.7959810  0.3196211  3.1246001  1.3829942
## [49]  1.4571013  1.7056487  0.8871131
```

```
low=murder_rate<1
print(low)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [13]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE
## [25] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
## [37] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE
## [49] FALSE FALSE  TRUE
```

(2) `which()` function results out the location of certain element. Used logical operations for filtering out the indices of which `murder_rate` is lower than 1. Result is shown on the code below.

```
indices=which(low)
indices
```

```
##  [1] 12 13 16 20 24 30 35 38 42 45 46 51
```

(3) Used object `indices` for filtering by square brackets.

Hawaii, Idaho, Iowa, Maine, Minnesota, New Hampshire, North Dakota, Oregon, South Dakota, Utah, Vermon, Wyoming.

```
state_name=murders[indices,'state']
state_name
```

```
##  [1] "Hawaii"        "Idaho"         "Iowa"          "Maine"
##  [5] "Minnesota"     "New Hampshire" "North Dakota"  "Oregon"
##  [9] "South Dakota"  "Utah"          "Vermont"       "Wyoming"
```

(4) Used logical class `condition` using &and ==.

Maine, New Hampshire, Vermont.

```
condition = low & murders$region=="Northeast"
murders$state[condition]
```

```
## [1] "Maine"         "New Hampshire" "Vermont"
```

```
# 특정 열에서 TRUE 값을 만족하는 데이터를 추출함 (논리연산으로 추출)
```

(5) 27 states are below average murder rate.

```
avg=mean(murder_rate)
length(which(murder_rate<avg))
```

```
## [1] 27
```

(6) AK for Alaska, MI for Michigan, IA for Iowa.

```
abbreviation=c("AK","MI","IA")
index=match(abbreviation,murders$abb) # 해당 abbreviation이 murders$abb 에서 몇번째 index 인지?
index
```

```
## [1]  2 23 16
```

```
murders$state[index]
```

```
## [1] "Alaska"   "Michigan" "Iowa"
```

(7) 'MA','ME','MI','MO' are the actual abbreviations, and 'others'MU' doesn't exist.

```
abbreviation2=c('MA','ME','MI','MO','MU')
abbreviation2 %in% murders$abb # 다시
```

```
## [1]  TRUE  TRUE  TRUE  TRUE FALSE
```

(8) 'MU' is not an actual abbreviation. Found out the index of `FALSE` by using `which()` and `!`operator.

```
index2=which(!abbreviation2 %in% murders$abb) # which는 TRUE의 index를 반환한다.
abbreviation2[index2] # 논리연산으로 추출
```

```
## [1] "MU"
```

3.6.

(7) Written function that for any given n computes Sn. Value when n=10 is "385".

```
compute_s_n=function(n){
  x=1:n
  sum(x^2)
}

compute_s_n(10)
```

```
## [1] 385
```

(8) Defined empty vector by `vector("numeric",25)`, and stored the results by for loop.

```
s_n=vector("numeric",25)
for (i in 1:25){
  s_n[i]=compute_s_n(i)
}
s_n
```

```
##  [1]    1    5   14   30   55   91  140  204  285  385  506  650  819 1015 1240
## [16] 1496 1785 2109 2470 2870 3311 3795 4324 4900 5525
```

(9) Used `sapply()` by using function `compute_s_n()`.

```
s_n=vector("numeric",25);
```

```r
s_n=sapply(1:25,compute_s_n)
print(s_n)
```

```
##  [1]    1    5   14   30   55   91  140  204  285  385  506  650  819 1015 1240
## [16] 1496 1785 2109 2470 2870 3311 3795 4324 4900 5525
```