Note 11. Input & Output

Introduction to Statistical Programming

Chanmin Kim

Department of Statistics
Sungkyunkwan University

2022 Spring

scan() Function

- scan():
  - It reads from a file or the keyboard and returns a vector object.
  - what= double(), character(), numeric() or logical() (Type of data to be read; default is double()).
  - sep=: If there is a separator in the file.

- Example for reading a file.

```
> x <- scan('dat1.txt')
Read 12 items
> x
 [1] 1 2 3 6 7 9 4 5 4 8 9 3

> x <- scan('dat2.txt')
Error in scan(file, what, nmax, sep, dec, quote, skip, nlines,
na.strings,  : scan() expected 'a real', got 'abc'

> x <- scan('dat2.txt',what=character())
Read 6 items
> x
[1] "3"   "7"   "8"   "6"   "5"   "abc"
```

scan() Function

- ```
  > x <- scan('dat2.txt',what=character(),sep='\n')
  Read 3 items
  > x
  [1] "3 7 8" "6 5"   "abc"
  ```

- Example for reading from the keyboard:
  - End of the input $\Rightarrow$ empty line.
- ```
  > x <- scan('')
  1: 1 2 3
  4: 3
  5: 4 5
  7:
  Read 6 items
  > x
  [1] 1 2 3 3 4 5
  ```

readline() Function

- readline(): Read a single line from the keyboard and return a character object.

```
> x <- readline()
abc efg
> x
[1] "abc efg"

> x <- readline()
1 2 3
> x
[1] "1 2 3"

> name <- readline('Type your name:  ')
Type your name:  JJ
> name
[1] "JJ"
```

Printing to Screen

- `print()`:
  - In interactive mode, you can print object values by typing object names.
  - To print objects in the body of functions, use `print()`.
  - This is a generic function.

```
> f <- function(x) print(x+10)
> f(10)
[1] 20
```

- `cat()`:
  - Concatenate and print outputs (multiple objects).
  - `\n`: End-of-line character. If you do not use this, the next call would continue to print on the same line.
  - If you want to distinguish outputs, `sep=` can be used.

Printing to Screen

```
> print('abc')
[1] "abc"
> cat('abc')
abc

> x <- 1:3
> cat(x,'abc','def\n')
1 2 3 abc def

> cat(x,'abc','def',sep='')
123abcdef

> cat(x,'abc','def',sep='\n')
1
2
3
abc
def
```

Printing to Screen

```
> cat(x,'abc','def\n',sep='.')
1.2.3.abc.def

> f <- function(x)  for (i in 1:3) cat(x+i)
> f(10)
111213

> f <- function(x)  for (i in 1:3) cat(x+i,'\n')
> f(10)
11
12
13
```

Reading a Data Frame from a File

- `read.table('data file', sep= , header=T)`:
  - It reads a data frame from a file.
  - *sep*= : If data in the file are separated by a certain sign or character, it can be specified by this argument.
  - *header*= : If there are variable names in the first line of the file, `header=T`. Otherwise, `header=F`.

```
> x <- read.table('dat3.txt',sep=',', header=T)
> x
  name                 dept score
1  Kim           Statistics    92
2  Lee          Mathematics    95
3 Park              English    90
4 Choi Applied Mathematics    91
```

- `readLines()`: Read a character vector from a file. It reads one line at a time.

```
> readLines('dat3.txt')
[1] "name,dept,score"  "Kim,Statistics,92"  "Lee,Mathematics,95"
[4] "Park,English,90"  "Choi,Applied Mathematics,91"
```

Reading a Data Frame from a File

- Using `scan()`.
- Using `read.table()` and `as.matrix()`

```
> x <- matrix(scan('dat1.txt'),4,byrow=T)
> x
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    6    7    9
[3,]    4    5    4
[4,]    8    9    3

> x <- as.matrix(read.table('dat1.txt'))
> x
     V1 V2 V3
[1,]  1  2  3
[2,]  6  7  9
[3,]  4  5  4
[4,]  8  9  3
```

Connection

- Connection:
  - Mechanism used in various kind of I/O operations.
  - Connection is created by calling `file()` (file access), `url()` (internet access), etc.

- `file('file name', open=)`
  - Read (`open='r'`) from a file and write (`open='w'`) to a file by `readLines()` and `writeLines()`, respectively.

```
> x <- file('dat3.txt','r')
> readLines(x,n=1)
[1] "name,dept,score"
> readLines(x,n=4)
[1] "Kim,Statistics,92" "Lee,Mathematics,95" "Park,English,90"
[4] "Choi,Applied Mathematics,91"
> readLines(x,n=1)
character(0)
> close(x)
```

Connection

```
> x <- file('dat3.txt','r')
> readLines(x,n=2)
[1] "name,dept,score"   "Kim,Statistics,92"
> seek(con=x, where=0)  # Back to the first line
[1] 36                  # File pointer is 36 now.
> readLines(x,n=1)
[1] "name,dept,score"
> close(x)

> x <- file('dat4.txt','w') # Create 'dat4.txt' file
> writeLines('abc\ndef',x)
> writeLines('ghl',x)
> close(x)
```

Connection

```
> # dat5.txt: column 1: ID;  column 2-3: age; # column 4-7: salary
> dt <- data.frame(stringsAsFactors = F)
> x <- file('dat5.txt','r')
> repeat
+ {
+   person = readLines(x,1)
+   if (length(person) == 0) break
+   l = vector('list',3)
+   names(l) = c('id','age','salary')
+   l$id = as.factor(substr(person,1,1))
+   l$age = as.numeric(substr(person,2,3))
+   l$salary = as.numeric(substr(person,4,7))
+   dt = rbind(dt,as.data.frame(l,stringsAsFactors = F))
+ }
> close(x);  dt
  id age salary
1  a  32   4200
2  b  21   5300
3  c  27   4300
4  d  35   3700
```

Read from a File in Web

- `read.table()` and `scan()` accepts web URL as arguments.

```
> x <- read.table('http://archive.ics.uci.edu/ml
/machine-learning-databases/iris/iris.data',sep=',')
> x[1:10,]
    V1  V2  V3  V4          V5
1  5.1 3.5 1.4 0.2 Iris-setosa
2  4.9 3.0 1.4 0.2 Iris-setosa
3  4.7 3.2 1.3 0.2 Iris-setosa
4  4.6 3.1 1.5 0.2 Iris-setosa
5  5.0 3.6 1.4 0.2 Iris-setosa
6  5.4 3.9 1.7 0.4 Iris-setosa
7  4.6 3.4 1.4 0.3 Iris-setosa
8  5.0 3.4 1.5 0.2 Iris-setosa
9  4.4 2.9 1.4 0.2 Iris-setosa
10 4.9 3.1 1.5 0.1 Iris-setosa
```

Writing a File

- `write.table(`*`data frame`*`,'data file')`: Write a data frame to a file.
- `cat('contents',file='file name')`: Create a file with contents.
- `cat('contents',file='file name',append=T)`: Append contents to the existing file.
- `writeLines('contents','w')`: Create a file with contents.
- `writeLines('contents','a')`: Append contents to the existing file.

Writing a File

```
> dt
  id age salary
1  a  32   4200
2  b  21   5300
3  c  27   4300
4  d  35   3700
> write.table(dt,'dat6.txt')

> cat('e253700\n',file='dat5.txt',append=T)

> x <- file('dat5.txt','a')
> writeLines('f256700',x)
close(x)
```