Chapter 6
Data Input and Output

Arthur Li

Creating Vectors By Using the `scan` Function

- **`scan`**: create a vector by typing the data directly into the console

```
> x = scan()
1: 1 3 5
4:
Read 3 items
> x
[1] 1 3 5
```

The reading will stop when a completely blank line is entered

- By default, **`scan`** expects all of its input to be numeric values

Creating Vectors By Using the `scan` Function

❖ To read character values, …

```
> char = scan(what = "")
1: a b c d e
6:
Read 5 items
> char
[1] "a" "b" "c" "d" "e"
```

Creating Matrices and Data Frames By Using the `scan` and `fix` Functions

❖ Use **`scan`** + **`matrix`** functions to read a data matrix

```
> mat = matrix(scan(), ncol = 2, byrow = TRUE)
1: 1 3
3: 3 4
5: 4 6
7:
Read 6 items
> mat
     [,1] [,2]
[1,]    1    3
[2,]    3    4
[3,]    4    6
```

Creating Matrices and Data Frames By Using the `scan` and `fix` Functions

```
> mat = matrix(scan(), ncol = 2)
1: 1 3
3: 3 4
5: 4 6
7:
Read 6 items
> mat
     [,1] [,2]
[1,]    1    4
[2,]    3    4
[3,]    3    6
```

Creating Matrices and Data Frames By Using the `scan` and `fix` Functions

❖ If **`what`** argument is a list → **`scan`** return a list of vectors

```
> dat = scan(what = list(name = "", height = 0, smoke = TRUE))
1: john 57 T
2: ken 50 F
3: dave 55 T
4:
Read 3 records
> dat
$name
[1] "john" "ken" "dave"
$height
[1] 57 50 55
$smoke
[1] TRUE FALSE TRUE
```

To specify numeric values

To specify logical values

Creating Matrices and Data Frames By Using the `scan` and `fix` Functions

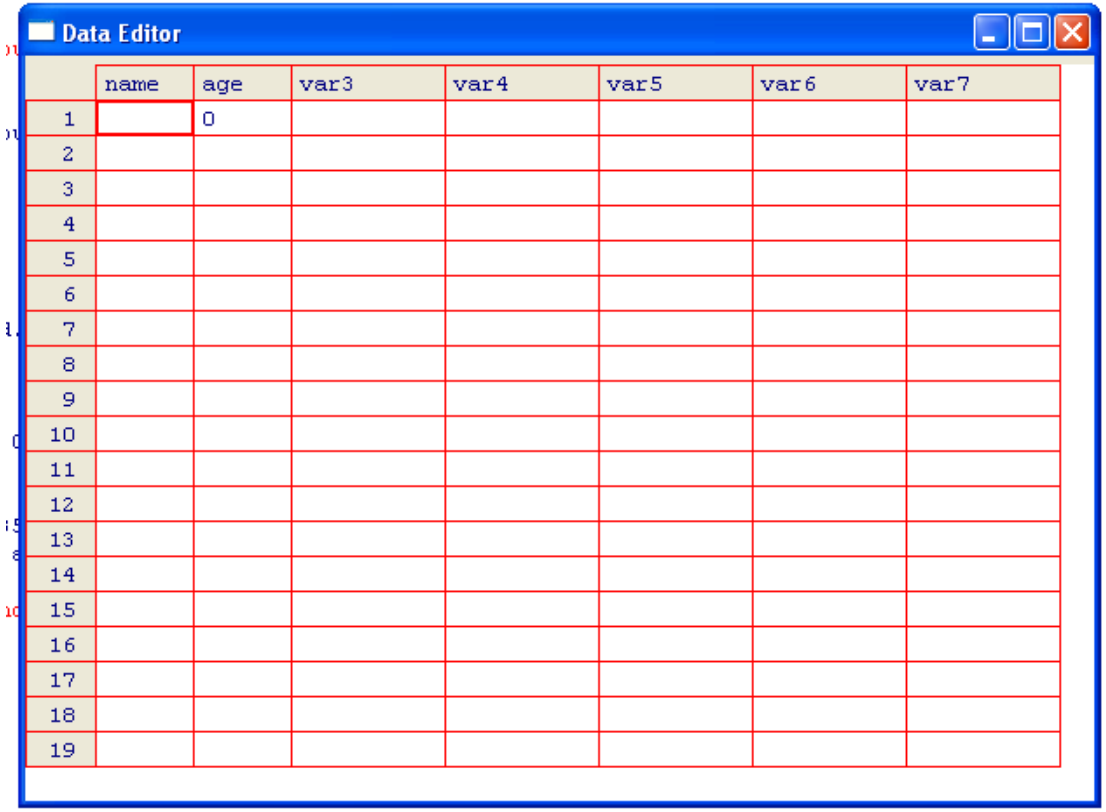
❖ Convert **`dat`** → a data frame by

```
> data.frame(dat)
  name height  smoke
1 john     57   TRUE
2  ken     50  FALSE
3 dave     55   TRUE
```

Creating Matrices and Data Frames By Using the `scan` and `fix` Functions

❖ The **`fix`** function:

```
> dummy = data.frame(name="", age=0)
> fix(dummy)
```



Reading White-space, Tab-, or Comma-separated Data

- **`read.table`**: read an external text file in which each field is separated by one or more separators

- The result is a data frame

- It has large number of arguments

Reading White-space, Tab-, or Comma-separated Data

- Before reading your file, consider the format criteria first
  - **`sep`**: default values – spaces, tabs, newlines
  - **`header`**: default values – FALSE → R use V1, V2, ...
  - header line is one column shorter than the body of the file, the 1st column → rownames. The header option is automatically set to TRUE
  - By default: **`read.table`** can recognize **`NA`** as a missing value for any data type; and treat **`NaN`**, **`Inf`**, and **`-Inf`** as missing for numeric data.
    `na.strings = "."` → Treat "." as missing values

Reading White-space, Tab-, or Comma-separated Data

- Before reading your file, consider the format criteria first

  - By default, any text after # are comments
    `comment.char ="%"` → text after % are comments

  - **`skip`**: To skip number of lines

  - **`nrow`**: To read number of rows

Reading White-space, Tab-, or Comma-separated Data

example1.txt:

```
##This file is borrowed from PM599: Programming In SAS
##Arthur Li
Fname    Lname    race    age     preg    income
KAREN    ARIAS    H
Caroline          Emb
GEN      ERECKSON
JOAN     RIVERA   W
ANDREA   Jones    B
BEVERLY  ROELL    W
LUISA    RUNYON   W
ELIZABETH/MARK SHA
KAREN    STANFIELD
LUCY     SWAIM    W
KATHY    ZACCAGNINO
Tami     Ogawa    A       18      0       39000
Rebecca  Chang    A       35      0       134000
Diana    Gonzales         H       26      1       29000
Angela   Xu       A       36      1       76000
CHRIS    DUDZINSKI        W       19      .       13900
```

- Each field in the data set is separated by a tab
- The first two lines are comments after the #
- The first row after the comments have the variable names
- The numerical missing values are represented as “.”

Reading White-space, Tab-, or Comma-separated Data

```
> setwd("C:/Users/Arthur/Documents/PM599 R Sp11/chapter4")
> example1 = read.table(file = "example1.txt", header = T,
+ na.strings = ".")

> head(example1)
     Fname    Lname race age preg income
1    KAREN    ARIAS    H  26    0  35000
2 Caroline   Embrey    W  26    1  48000
3      GEN ERECKSON    W  32    1  30000
4     JOAN   RIVERA    W  17    0  59000
5   ANDREA    Jones    B  29    1 120000
6  BEVERLY    ROELL    W  26    1 113000
```

- **`header = T`**
- **`comment.char`** argument is not used
- **`na.strings = "."`**
- **`sep`** option is not used. Or **`sep= "\t"`**

Reading White-space, Tab-, or Comma-separated Data

```
> class(example1$Fname)
[1] "factor"
```

❖ To prevent conversion to factors, you can set **`stringsAsFactors`** to FALSE

```
> example1 = read.table(file = "example1.txt", header = T,
  + na.strings = ".", stringsAsFactors = F)
> class(example1$Fname)
[1] "character"
```

Writing R Objects to Text Files
The `write` function

- **`write:`** usually used to write a matrix to a file

- Three important arguments:
  - **`x`**: The data to be written-out
  - **`file`**: The name of the output file
  - **`ncolumns`**: The number of columns to write to the output data. By default, the **`write`** function writes the **`x`** to the output file in 5 columns and stored by columns

The `write` function

```
> foo = matrix(1:18, ncol = 3)
> foo
     [,1] [,2] [,3]
[1,]    1    7   13
[2,]    2    8   14
[3,]    3    9   15
[4,]    4   10   16
[5,]    5   11   17
[6,]    6   12   18
> write(foo, "foo.txt", ncol = 3)
```

foo.txt

```
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
16 17 18
```

The `write` function

```
> foo1 = t(foo)
> foo1
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    3    4    5    6
[2,]    7    8    9   10   11   12
[3,]   13   14   15   16   17   18
> write(foo1, "foo1.txt", ncol = 3)
```

foo1.txt

```
1 7 13
2 8 14
3 9 15
4 10 16
5 11 17
6 12 18
```

The `write.table` function

❖ **`write.table`**: write a data frame/matrix to an output file

❖ The most common arguments for the **`write.table`**:

- ❑ **`file`**: The name of the output file
- ❑ **`quote`**:
  - ✓ TRUE - (the default value), any character / factor columns will be surrounded by double quotes
  - ✓ FALSE - all the quotes will be eliminated
  - ✓ a numeric vector - serves as the indices of columns to quote
- ❑ **`sep`**: e.g. **`sep = "\t"`**
- ❑ **`na`**: the default value is **`NA`**
- ❑ **`row.names`**: By default, the rownames will be printed
- ❑ **`col.names`**: By default, the colnames will be printed

The `write.table` function

```
> dat1 = data.frame(numVar = c(round(rnorm(5), 2), NA),
+ charVar = c(NA, letters[1:5]))
> dat1
  numVar charVar
1  -0.38    <NA>
2   0.32       a
3  -0.32       b
4  -0.06       c
5  -0.71       d
6     NA       e
> write.table(dat1, file = "dat1.txt")
```

dat1.txt

```
"numVar" "charVar"
"1" -1.66 NA
"2" -1.36 "a"
"3" 0.79 "b"
"4" 0.27 "c"
"5" -0.34 "d"
"6" NA "e"
```

- Each column is separated by one space
- Character values are quoted

The `write.table` function

```
> write.table(dat1, file = "dat1.txt", row.names = F,
+ quote = F, sep = "\t", na = "")
```

dat1.txt

```
numVar charVar
1.16
-0.76  a
1.12   b
0.41   c
-0.2   d
  e
```

- you can easily open this file by using EXCEL