

Stat 302
Statistical Software and Its Applications
Data Import and Export

Yen-Chi Chen

Department of Statistics, University of Washington

Autumn 2016

General Remarks on I/O (Import/Export)

- We generally don't type data manually into R.
- In earlier times data came to us in ASCII files.
ASCII = American Standard Code for Information Interchange.
- Items had to be in some logical order so that they could be arranged properly for use in software.
- Data items were separated by spaces or tabs (`\t`) or other special characters that are not used as part of data items.
- Nowadays the most common form of a data file is a `filename.csv` file as provided by Excel or OpenOffice Calc.
`csv` = comma-separated values.
- Many software platforms can export and import data formats for other platform, e.g., $R \longleftrightarrow \text{SAS}$.
- After I/O verify that the data are in the correct form.

Importing from an ASCII Text File

- The first 3 and last line of the ASCII text file `ReactionTime.txt` are given below.
- The first line provides the header information for the variables.
- For each EMS¹ call these variables are the reaction time to get out the fire station door, the fire station, and the crew shift.

```
Reaction, Station, Shift  
86, ST64, B  
182, ST64, B  
...  
189, ST64, B
```

¹Emergency Medical Service

Using read.table

```
> ReactionTime <- read.table("ReactionTime.txt",
                             sep="," ,header=T)

> str(ReactionTime)
'data.frame':  2028 obs. of  3 variables:
 $ Reaction: int  86 182 132 196 160 3 0 0 95 152 .
 $ Station  : Factor w/ 3 levels "ST63","ST64",...: 2
 $ Shift    : Factor w/ 4 levels "A","B","C","D": 2

> ReactionTime[c(1,2,2028),]
      Reaction Station Shift
1           86     ST64     B
2          182     ST64     B
2028        189     ST64     B
```

- You need to specify the location of the file for the `read.table` function.
- Use `getwd()` to check current directory address.
- Use `setwd("<location of your dataset>")` to change your directory address.
- Or you can use *Session* \implies *Set Working Directory* \implies *Choose Directory* to change.

Features of `read.table`

- It imports the data as a `data.frame`.
- There are many arguments in `read.table`.
- Generally we use only the first three arguments.
- The first argument `file` is the location of the file.
- The second argument `header` is a logical argument; `TRUE` if the file contains a header for each variable.
- The third argument `sep` is how the variables for one observation are separated.

read.table: header - 1

```
> is.data.frame(ReactionTime)
[1] TRUE
> ReactionTime_1 <- read.table("ReactionTime.txt",
+                               header=F, sep=",")
> head(ReactionTime_1)
```

	V1	V2	V3
1	Reaction	Station	Shift
2	86	ST64	B
3	182	ST64	B
4	132	ST64	B
5	196	ST64	B
6	160	ST64	A

```
> ReactionTime_1 <-  
+   read.table("ReactionTime_no_header.txt",  
+             header=F, sep=",")  
>  
> head(ReactionTime_1)  
  V1  V2 V3  
1  86 ST64 B  
2 182 ST64 B  
3 132 ST64 B  
4 196 ST64 B  
5 160 ST64 A  
6   3 ST65 A  
> # if the file has no header: OK
```


read.table: header - 3

```
> ReactionTime_1 <-  
+   read.table("ReactionTime_no_header.txt",  
+             header=T, sep=",")  
>  
> head(ReactionTime_1)  
  X86 ST64 B  
1 182 ST64 B  
2 132 ST64 B  
3 196 ST64 B  
4 160 ST64 A  
5   3 ST65 A  
6   0 ST64 A  
> # if no header but header=T: the first  
> # observation is treated as the header
```

read.table: sep

```
> ReactionTime_1 <- read.table("ReactionTime.txt",
+                               header=T, sep="")
>
> head(ReactionTime_1)
  Reaction.Station.Shift
1           86,ST64,B
2          182,ST64,B
3          132,ST64,B
4          196,ST64,B
5          160,ST64,A
6           3,ST65,A
> # the wrong "sep" will merge variables...
> length(ReactionTime_1)
[1] 1
```

Importing from a .csv File

- It is possible to import directly from an Excel spreadsheet, but the advice is to convert it to a [single](#) sheet .csv file.
- The first 3 and last line of the file ReactionTime.csv are shown below

ReactionTime.csv - OpenOffice.org Calc

File Edit View Insert Format Tools Data Window Help

Liberation Sans 10

A4:AMJ2028 $f(x)$ Σ = 67

	A	B	C	D	E	F	G	H
1	Reaction	Station	Shift					
2	86	ST64	B					
3	182	ST64	B					
2029	189	ST64	B					

Using read.csv

```
> ReactionTimecsv <- read.csv("ReactionTime.csv",  
+                               header=T, sep=",")  
>  
> head(ReactionTimecsv)
```

	Reaction	Station	Shift
1	86	ST64	B
2	182	ST64	B
3	132	ST64	B
4	196	ST64	B
5	160	ST64	A
6	3	ST65	A

Features of `read.csv`

- If variables are separated by semicolon `;`, then use `read.csv2`.
- `read.csv` and `read.csv2` are identical to `read.table` except for the defaults.
- In addition to `read.csv`, there are many other data import approach based on `read.table`.
- Try `help(read.table)`.

Using write.table

```
> write.table(ReactionTime, "xx.txt", sep=",")
```

```
# produces a file xx.txt with first 3 lines
```

```
"Reaction", "Station", "Shift"
```

```
"1", 86, "ST64", "B"
```

```
"2", 182, "ST64", "B"
```

Read the documentation on `write.table` and `read.table`.

Features of `write.table`

- There are many arguments in `read.table`.
- Here are some commonly used arguments:
 - `x`: the data you want to export.
 - `file`: the file name (and location) you want to output to.
 - `quote`: adding quotes to the character/factor variables.
 - `sep`: how different variables are separated.
 - `row.names/col.names`: output the name of row/the name of column.
- Read the documentation on `write.table` and `read.table`.

Using write.table: quote

```
> write.table(ReactionTime, "xx.txt", sep=",",  
              quote=F)  
# produces a file xx.txt with first 3 lines
```

```
Reaction, Station, Shift  
1, 86, ST64, B  
2, 182, ST64, B
```


Using write.table: row.name

```
> write.table(ReactionTime, "xx.txt", sep=",",  
+             quote=F, row.name=F)
```

```
# produces a file xx.txt with first 3 lines
```

```
Reaction, Station, Shift  
86, ST64, B  
182, ST64, B
```

Using write.table: col.name

```
> write.table(ReactionTime, "xx.txt", sep=",",  
+             quote=F, row.name=F)
```

```
# produces a file xx.txt with first 3 lines
```

```
1, 86, ST64, B
```

```
2, 182, ST64, B
```

```
3, 132, ST64, B
```

Using write.table: sep - 1

```
> write.table(ReactionTime, "xx.txt", sep=" ",  
+           quote=F)
```

```
# produces a file xx.txt with first 3 lines
```

```
Reaction Station Shift
```

```
1 86 ST64 B
```

```
2 182 ST64 B
```

Using write.table: sep - 2

```
> write.table(ReactionTime, "xx.txt", sep="--",  
+             quote=F)
```

```
# produces a file xx.txt with first 3 lines
```

```
Reaction--Station--Shift  
1--86--ST64--B  
2--182--ST64--B
```

Manipulating Data – 1

We can do some analysis and add extra columns to the original data.

```
> new_time <- ReactionTime$Reaction/10
>
> ReactionTime_new <- cbind(ReactionTime, new_time)
>
> colnames(ReactionTime_new)[4] = "RT_mins"
>
> head(ReactionTime_new)
  Reaction Station Shift RT_mins
1       86   ST64    B     8.6
2      182   ST64    B    18.2
3      132   ST64    B    13.2
> write.table(ReactionTime_new, "new_RT.txt",
+             sep=" ", quote=F, row.names=F)
```

Manipulating Data – 2

We may remove columns or rows in the data.

```
> ReactionTime_rm <- ReactionTime[,-3]
> # this removes the third column.
> head(ReactionTime_rm)
  Reaction Station
1      86     ST64
2     182     ST64
3     132     ST64
4     196     ST64
5     160     ST64
6       3     ST65
```

Summarizing Data – 1

`summary(x)`: a function that provides summary information for the object `x`.

```
> summary(ReactionTime)
  Reaction      Station      Shift
Min.   :  0.0    ST63:498    A:477
1st Qu.: 73.0    ST64:902    B:531
Median : 99.0    ST65:628    C:479
Mean   :100.3
3rd Qu.:132.0
Max.   :318.0
```

Summarizing Data – 2

`table(x)`: a function that creates a table for summarizing information. Particularly useful for objects whose structure is factor.

```
> table(ReactionTime$Station)
```

```
ST63 ST64 ST65  
 498  902  628
```

```
>
```

```
> table(ReactionTime$Station, ReactionTime$Shift)
```

```
      A    B    C    D  
ST63 120 133 117 128  
ST64 205 216 221 260  
ST65 152 182 141 153
```


Built-in Datasets in R

R has many built-in datasets. Try `data()`.

```
> data()
```

```
>
```

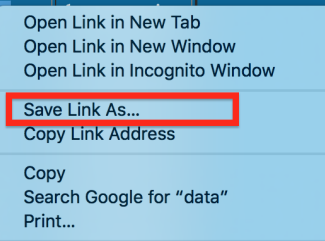
```
> head(iris)
```

```
> # 'iris' is a well-known dataset in R.
```

Getting Data from the Internet

- In many cases, you can download the data by clicking the link.
- But sometimes you may need to *right click* the link and choose *Save Link As*
- As an example, go to <http://www.stat.cmu.edu/~larry/all-of-nonpar/data.html> and try to download a dataset:

Section first mentioned	Data file	Documentation
Example 2.2	da	
Exercise 2.11	da	
Exercise 2.12	da	
Exercise 2.13	da	
Example 4.1	da	
Example 4.2	da	
Example 4.3	da	



A screenshot of a right-click context menu over a link in the table. The menu is light blue and contains the following options: 'Open Link in New Tab', 'Open Link in New Window', 'Open Link in Incognito Window', 'Save Link As...' (highlighted with a red rectangle), 'Copy Link Address', 'Copy', 'Search Google for "data"', and 'Print...'. The link being right-clicked is the first one in the table, 'Example 2.2'.

Interfacing with other Packages

- R has a base package called `foreign`, that interfaces with other packages, such as SPSS, SAS, Systat, Octave, Stata, Minitab.
- To activate its commands you first have to issue the command `library(foreign)`.
- Under `help.start()`, html help interface, read the documentation for functions in `foreign`.
- They mostly concern reading exports from these other packages for use in R.

- Packages are available to connect to some databases such as
 - MySQL
 - Oracle
 - PostgreSQL
 - SQLite
- Consult Chapter 4 of the R manual R-Data Import/Export.
- Consult this manual on all other issues concerning I/O.

- Previously we referred to the package `foreign`.
- To use any function in it you need to execute `library(foreign)`.
- That works because `foreign` is in the base distribution of R.
- For other packages (there are over 9000) you need to first install it on your system via `install.packages("packageName")`.
- When prompted choose a distribution site near you.
- You only need to do this install step once.
- The `library` command needs to be done anew for each new R session that wants to use the package functions.

- `cars` is a built-in dataset in R.
- To explore this dataset, try `summary(cars)` and `head(cars)`.
- Now declare two new variables by

```
speed2 <- cars$speed^1.5
res <- speed2-cars$dist
```
- Create a new data.frame whose first two columns are the same as `cars` and the third column is the vector `speed2` and the last column is the vector `res`.
- Output the dataset with file name `cars_new.txt`.