Reading and writing files

Importing data in R

Data contained in external text files can be imported in R using one of the following functions:

- scan()
- read.table()
- read.csv()
- read.csv2()
- read.delim()
- read.delim2()

The requirements on the form of data set are sometimes quite strict, so it is better to modify input files to satisfify this requirements.

The function scan()

This function is the most flexible: it can be used to read logical, integer, numeric, complex, character, raw data and *lists*

```
scan(file = " ", what = double(0), n = -1, sep = "", dec =
    ".", skip = 0, na.strings = "NA")
```

file: the name of the file which the data are to be read from;

what: the type of data to be read (logical, integer, numeric, complex, character, raw and list);

n: the maximum number of data values to be read, defaulting to no limit;

scan(file = "", what = double(0), n = -1, sep = "", dec = ".", skip = 0, na.strings = "NA")

sep: the field separator character. If sep = “”, the separator is ‘white space’, that is one or more spaces, tabs;

dec: the character used in the file for decimal points;

skip: the number of lines of the input file to skip before beginning to read data values;

na.strings: character vector. Elements of this vector are to be interpreted as missing (NA) values. Blank fields are also considered to be missing values in logical, integer, numeric and complex fields

The function read.table()

This function reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

read.table(file, header = FALSE, sep = "", dec = ".", row.names, col.names)

header: a logical value indicating whether the file contains the names of the variables as its first line

sep: the field separator character. If sep = "" (the default for read.table) the separator is ‘white space’, that is one or more spaces, tabs;

dec: the character used in the file for decimal points;

row.names: a vector of row names;

col.names: a vector of column names.

Input file format

Row names

Column names

	N.Amer	Europe	Asia	S.Amer	Oceania	Africa	Mid.Amer
1951	45939	21574	2876	1815	1646	89	555
1956	60423	29990	4708	2568	2366	1411	733
1957	64721	32510	5230	2695	2526	1546	773
1958	68484	35218	6662	2845	2691	1663	836
1959	71799	37598	6856	3000	2868	1769	911
1960	76036	40341	8220	3145	3054	1905	1008
1961	79831	43173	9053	3338	3224	2005	1076

```
a <-read.table("Worldtelephones.txt")
>str(a)
'data.frame':   7 obs. of  7 variables:
 $ N.Amer  : int  45939 60423 64721 68484 71799 76036 79831
 $ Europe  : int  21574 29990 32510 35218 37598 40341 43173
 $ Asia    : int  2876 4708 5230 6662 6856 8220 9053
 $ S.Amer  : int  1815 2568 2695 2845 3000 3145 3338
 $ Oceania : int  1646 2366 2526 2691 2868 3054 3224
 $ Africa  : int  89 1411 1546 1663 1769 1905 2005
 $ Mid.Amer: int  555 733 773 836 911 1008 1076
```

If only column labels are present add the option "header=T"

The function read.csv() and read.csv2()

- **read.csv()** is intended for reading 'comma separated value' (CSV) files, where the decimal point is "."

- **read.csv2()** is the variant used in countries that use a comma (",") as decimal point and a semicolon (";") as field separator.

```
read.csv(file, header = TRUE, sep = ",", dec=".“)
read.csv2(file, header = TRUE, sep = ";",  dec=",“)
```

The functions read.delim() and read.delim2()

- They are intended to read TAB separated files

```
read.delim(file, header = TRUE, sep = "\t", dec=".", fill =
    TRUE, ...)

read.delim2(file, header = TRUE, sep = "\t", dec=",", fill =
    TRUE,...)
```

sep: the field separator character. “\t” (default for read.delim) stands for TAB separator;

fill: if TRUE then in case the rows have unequal length, blank fields are implicitly added

Exporting Data

R objects can be exported to a text file using the **cat()** function:

```
cat (x , file = "", sep = " ", fill = FALSE, labels = NULL,
    append = FALSE)
```

x: **R** object

file: character string naming the file to print to. If "" (the default), cat prints to the standard output connection, the console unless redirected by sink.

sep: a character vector of strings to append after each element.

fill: controls how the output is broken into successive lines.

append:logical. If TRUE output will be appended to file; otherwise, it will overwrite the contents of file.

Writing data frames

Often it is useful to write a matrix or a data frame to a file as a grid of elements.

- write() writes out a matrix or vector in a specified number of columns.

- write.table() writes out a data frame (or an object that can be coerced to a data frame) with row and column labels

The function write()

write (x, file = "data", ncolumns =, append = FALSE,
sep = " ")

x	the data to be written out.
file	the file to write to
ncolumns	the number of columns to write the data in.
append	if TRUE the data x are appended to the file.
sep	a string used to separate columns. Using sep = "\t" gives tab delimited output; default is white space " ".

The function write.table()

```
write.table(x, file = "", append = FALSE, sep = " ", na =
    "NA", dec = ".", row.names = TRUE, col.names = TRUE)
```

na:

the string to use for missing values in the data (default is NA)

row.names (col.names):

either a logical value indicating whether the row (column) names of x are to be written along with x, or a character vector of row (column) names to be written.

The functions write.csv() and write.csv2()

write.csv(x, file=“ ”)

write to a file using the comma (“,”) as the field separator

write.csv2(x, file=“ ”)

write to a file using semicolon (“;”) as the field separator and the comma as the decimal point

Reading Excel files

An excel file can contain many sheets, and the sheets can contain formulae, macros and so on: not all readers can manage this.

Instead of importing the data contained in .xls files in R, it would be easier if:

- you export data of the .xls file in a .txt file, in tab-delimited or comma-separated form
- you use R functions *read.delim()* or *read.csv()* to import the .txt file into the R workspace