

# Writing data to text files

Programming in R for Data Science

Anders Stockmarr, Kasper Kristensen, Anders Nielsen

# Data Import

Just like data import, data can be exported from R can import data many ways. You can export directly to

- ▶ EXCEL;
- ▶ Plain text files;
- ▶ SAS;
- ▶ SPSS;
- ▶ STATA;
- ▶ etc.

One package to use in case of SAS, SPSS and STAT is the `foreign` package.

As with data import, issues that you must attend to is in most cases similar.

We shall consider issues when exporting to plain text files.

## Writing data to a text file

Data frame to be written:

```
> mydat<-data.frame(x1=c(2,2,3,3,1,1),  
+                   x2=c(0.3,1,2.1,2.2,0.1,0.2),  
+                   x3=c(0.01,0.11,0.04,0.02,0.1,0.06))
```

```
> mydat
```

	x1	x2	x3
1	2	0.3	0.01
2	2	1.0	0.11
3	3	2.1	0.04
4	3	2.2	0.02
5	1	0.1	0.10
6	1	0.2	0.06

The function to use is `write.table()`; the reverse of `read.table()`.

## Writing data to a text file

Basic write command:

```
write.table(mydat, file="c:/datadir/filename.dat",  
            row.names=FALSE, col.names=FALSE, sep=", ")
```

Example:

- ▶ Lets write the mydat object to the file write.datatest.txt, in the folder Data in the R working directory.
- ▶ We need only specify the path from the working directory.

```
write.table(mydat, file="Data/write.datatest.txt",  
            row.names=FALSE, col.names=FALSE, sep=", ")
```

## Variants of write.table()

- ▶ useful variants of write.table are:
  - ▶ `write.csv()` comma separated, dot as decimal point
  - ▶ `write.csv2()` `sep=";"` and `dec=","`
- ▶ Additional arguments are similar to those of `write.table()`

files saved with both `write.csv()` and `write.csv2()` can be entered into Excel as plain text.

## Basic writing functions

`cat()`

```
> cat("Test file for cat\n",round(rnorm(5),3),"\n",  
+     file="cattest.txt")
```

```
| Test file for cat  
| -1.473 -0.088 1.551 0.217 1.417  
|
```

`writeLines()`

```
> lin<-c("Count down", paste(rev(1:10), collapse="-"),  
+       "Go")  
> writeLines(lin, con="Data/writelinetest.txt")
```

```
| Count down  
| 10-9-8-7-6-5-4-3-2-1  
| Go  
|
```

## A few special ones: `sink()`

`sink()`:

```
> sink("Data/sinktest.txt")
> x<-1:5
> y<-1:3
> outer(x,y)

      [,1] [,2] [,3]
[1,]     1     2     3
[2,]     2     4     6
[3,]     3     6     9
[4,]     4     8    12
[5,]     5    10    15

> sink()
```

output is not echoed in the R command prompt

## A few special ones: dump()

dump():

```
> x<-1:3  
> y<-rpois(10, 4)  
> dump(c("x","y"), file="Data/dumptest.txt")
```

```
| x <-  
| 1:3  
| y <-  
| c(2L, 2L, 3L, 4L, 5L, 1L, 6L, 3L, 3L, 1L) |
```

"L" signifies that the number is an integer in R:

```
> (2==2L)
```

```
[1] TRUE
```

Dumped files can be *sourced* with the source() command.



## A few special ones: dput()

dput():

```
> lis<-list(x=1:5, y=3, z=c("a","b","c"))  
> dput(lis, file="Data/dputtest.txt")  
| structure(list(x = 1:5, y = 3, z = c("a", "b", "c")), .Names = c("x",  
| "y", "z")) |
```

dget() inverses dput(). Note that the dget() command below doesn't restore lis, but creates an object similar to lis, which can be assigned to other objects:

```
> dget("Data/dputtest.txt")
```

```
$x
```

```
[1] 1 2 3 4 5
```

```
$y
```

```
[1] 3
```

```
$z
```

```
[1] "a" "b" "c"
```

## Using file connections

```
> f2<-file("Data/testout.txt", open="w")  
> cat("Header of file\n\n", file=f2)  
> mat<-matrix(round(rnorm(12),8), ncol=3)  
> write.table(mat, file=f2, row.names=FALSE, col.names=FALSE)  
> close(f2)
```

Header of file		
-1.59259661	0.33184189	-0.03059243
-0.14587929	0.37590371	0.46158609
-1.00489266	0.28096447	0.97548552
-1.53377867	-0.60433386	-0.48015229

## Using append=TRUE

```
> cat("Header of file\n\n", file="Data/testappend.txt")
> mat<-matrix(round(rnorm(12),8), ncol=3)
> write.table(mat, file="Data/testappend.txt", row.names=FALSE,
+             col.names=FALSE, append=TRUE)
```

```
Header of file
-1.28961805 0.52742282 0.19230131
0.67866701 0.38493936 -0.94428386
-0.39750542 -0.98370642 -1.26234264
0.43367006 -0.77599255 -0.22458311
```

## Working with binary files: Using save() and load()

- ▶ R also has its own internal binary format
- ▶ To save data and functions to it use e.g:

```
> x<-rnorm(3)
> lis<-list(y=1:5, z="lalala", fun=function()cat("ha-ha-ha\n"))
> save(x,lis, file="Data/test1.RData")
```
- ▶ To read back into R simple use:

```
> rm(list=ls())
> load(file="Data/test1.RData")
> ls()

[1] "lis" "x"
```
- ▶ This format is much simpler to use, but only works within **R** .