

How do I read data into R?

There is no such thing as an R system file similar to a Stata `.dta` or an SPSS `.sav` file. Instead, R reads data from a variety of formats – including files created in other statistical packages – directly into working memory. R generally lacks intuitive commands for data management, so users typically prefer to clean and prepare data with SAS, Stata, or SPSS. Once the data are ready, several functions are available for getting the data into R.

Reading Data Files in SPSS, Stata, and SAS formats

The **foreign** package can be used to read data stored as SPSS `.sav` files, Stata `.dta` files, or SAS XPORT libraries. If **foreign** is not already installed on your local computer, go to the **Packages** menu and choose **Install package(s)**.

If prompted, choose the closest CRAN mirror. When the **Packages** dialog box appears, scroll down to choose `foreign` and then click **OK**.

To use the commands in **foreign** one must first attach the library using the `library` function. At the prompt, type

```
> library(foreign)
```

As an example of reading data from other formats, assume that there is an SPSS file called **survey.sav** saved in the directory `C:\mydata`. The `read.spss` function from the **foreign** library will read the file into R.

```
> dataSPSS<-read.spss("C:/mydata/survey.sav", to.data.frame=TRUE)
```

This creates a data object called `dataSPSS` that is ready for analysis. The `to.data.frame` argument, whose default value is **FALSE**, tells R to treat the object as a data frame. Note that when specifying the pathname, R understands forward slashes whereas Windows reads backward slashes. If it is necessary to read in several data files from the same directory, the amount of typing can be reduced by first setting the working directory and then using the relative pathname. For example,

```
> setwd("C:/mydata")
```

```
> dataSPSS<-read.spss("survey.sav", to.data.frame=TRUE)
```

Alternatively, if one prefers to search for the location of a data file, one can type

```
> dataSPSS<-(file.choose(), to.data.frame=TRUE)
```

This will open a dialog box that can be used to navigate to the appropriate folder.

R will assume that any value labels recorded in the SPSS file refer to factors (categorical variables) and will store the labels rather than the original number. For example, a variable named `gender` may be coded `0=male` and `1=female`, and the labels are saved in the `.sav` file. When R reads in the data from SPSS, the values of the variable will be "male" and "female" rather than "0" and "1". This is the default behavior, but it can be changed in the call to the `read.spss` function:

```
> dataSPSS<-read.spss(file.choose(), use.value.labels=FALSE)
```

Reading Stata files is equally straightforward using the `read.dta` function. Assuming there is a Stata data file `survey.dta` in the `C:\mydata` folder, the appropriate syntax is

```
> dataStata<-read.dta("C:/mydata/survey.dta")
```

or

```
> dataStata<-read.dta(file.choose())
```

The created object is automatically a data frame. The default is to convert value labels into factor levels ("male" and "female" rather than "0" and "1"), but this can be turned off.

```
> dataStata<read.dta(file.choose(), convert.factors=FALSE)
```

Note that Stata sometimes changes how it stores data files from one version to the next, and the `foreign` package may lag a little behind. If the `read.dta` command returns an error, try saving the data in Stata using the `.saveold` command. This will create a `.dta` file saved in a previous version of Stata that `read.dta` may be more likely to recognize.

R can also read SAS XPORT libraries. The function takes only a single argument, the pathname:

```
> dataXPORT<-read.xport("C:/mydata/survey")
```

The function returns a data frame if there is a single dataset in the library or a list of data frames if there are multiple datasets.

Reading in ASCII files

R can also easily read in space-, tab-, and comma-delimited text files. The `read.table` function handles the first two cases; `read.csv` handles the other. Say there is an ASCII data file `survey.dat` in which white space separates the values for each variable. The following syntax reads in this data.

```
> dataTEXT<-read.table("C:/mydata/survey.dat", header=TRUE, sep= " ")
```

The `header` argument tells R that the first row includes variable names. Its default is `FALSE`. The `sep` argument specifies that values are separated by any white space, which is the default. If the values are separated by tabs, the value of the `sep` argument is changed to

```
> dataTAB<-read.table("C:/mydata/survey.dat", header=TRUE, sep= "\t")
```

The `read.csv` command is available for reading data files with comma-separated values.

```
> dataCOM<-read.csv("C:/mydata/survey.csv", header=TRUE)
```

The following are also equivalent:

```
> setwd("C:/mydata") > dataCOM<-read.csv("survey.csv", header=TRUE)
```

and

```
> dataCOM<-read.csv(file.choose(), header=TRUE)
```

It is also possible to read fixed format ASCII files – those with pre-specified columns and no delimiters – using the `read.fwf` function. However, this task is tedious (as it is in any package). For ICPSR data it is recommended to use the available setup files to read fixed format data into another package and then use the commands in R's `foreign` library.

Data in Excel Format

The easiest way to get Excel data into R is to save the spreadsheet as a comma-separated file and use R's `read.csv` function. The file type can be altered in Excel by changing the **Save as type** option to **CSV (Comma Delimited)**.



This work is licensed under a [Creative Commons Attribution-Noncommercial 3.0 United States License](https://creativecommons.org/licenses/by-nc/3.0/us/).