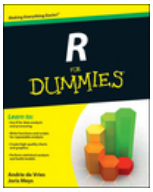
dummies
A Wiley Brand

HOW TO ROUND OFF NUMBERS IN R



RELATED BOOK

R For Dummies

By **Andrie de Vries, Joris Meys**

Although R can calculate accurately to up to 16 digits, you don't always want to use that many digits. In this case, you can use a couple functions in R to round numbers. To round a number to two digits after the decimal point, for example, use the round() function as follows:

```
> round(123.456,digits=2)
[1] 123.46
```

You also can use the round() function to round numbers to multiples of 10, 100, and so on. For that, you just add a negative number as the digits argument:

```
> round(-123.456,digits=-2)
[1] -100
```

If you want to specify the number of significant digits to be retained, regardless of the size of the number, you use the signif() function instead:

```
> signif(-123.456,digits=4)
[1] -123.5
```

Both round() and signif() round numbers to the nearest possibility. So, if the first digit that's dropped is smaller than 5, the number is rounded down. If it's bigger than 5, the number is rounded up.

If the first digit that is dropped is exactly 5, R uses a rule that's common in programming languages: Always round to the nearest even number. round(1.5) and round(2.5) both return 2, for example, and round(-4.5)

returns -4.

Contrary to round(), three other functions always round in the same direction:

- floor(x) rounds to the nearest integer that's smaller than x. So floor(123.45) becomes 123 and floor(-123.45) becomes –124.
- ceiling(x) rounds to the nearest integer that's larger than x. This means ceiling (123.45) becomes 124 and ceiling(123.45) becomes –123.
- trunc(x) rounds to the nearest integer in the direction of 0. So trunc(123.65) becomes 123 and trunc(-123.65) becomes –123.

CHEAT SHEET

R FOR DUMMIES CHEAT SHEET

From **R For Dummies, 2nd Edition**

By **Andrie de Vries, Joris Meys**

R is more than just a statistical programming language. It's also a powerful tool for all kinds of data processing and manipulation, used by a community of programmers and users, academics, and practitioners. But in order to get the most out of R, you need to know how to access the R Help files and find help from other sources. To represent data in R, you need to be able to succinctly and correctly specify subsets of your data. Finally, R has many functions that allow you to import data from other applications.

GETTING HELP WITH R

Even with good introductory books on R, you'll need to use the R Help files. The R Help files provide detailed information about the use of different functions and their peculiarities. R has excellent built-in help for every function that explains how to use that function. Just about every Help page has some examples that demonstrate how to use that function.

To search through the Help files, you'll use one of the following functions:

- ?: Displays the Help file for a specific function. For example, ?data.frame displays the Help file for the data.frame() function.
- ??: Searches for a word (or pattern) in the Help files. For example, ??list returns the names of functions that contain the

word list in either the function names or their descriptions.

- RSiteSearch(): Performs an online search of **RSiteSearch**. This search engine allows you to perform a search of the R functions, package vignettes and the R-help mail archives. For example, RSiteSearch("linear models") does a search at this website for the search term "linear models."

You aren't limited to the R Help files if you're looking for help with R. The add-on package sos, available for download from CRAN **here**, has some neat functions to search all the Help files on **RSiteSearch**. It displays results in a web browser window, making it easy to work with.

To use the package sos, you need to install the package by typing **install.packages("sos")** in your R console, and then load the package with library("sos").

Then you can use the findFn() function to do your search. For example, by typing **findFn("regression")** into your R console, you get a web page with the names, descriptions and links to several hundred functions that contain the word *regression* in the function name or Help text description.

IMPORTING DATA INTO R

R has many functions that allow you to import data from other applications. The following table lists some of the useful text import functions, what they do, and examples of how to use them.

Function	What It Does	Example

read.table()	Reads any tabular data where the columns are separated (for example by commas or tabs). You can specify the separator (for example, commas or tabs), as well as other arguments to precisely describe your data.	read.table(file="myfile", sep="t", header=TRUE)
read.csv()	A simplified version of read.table() with all the arguments preset to read CSV files, like Microsoft Excel spreadsheets.	read.csv(file="myfile")
read.csv2()	A version of read.csv() configured for data with a comma as the decimal point and a semicolon as the field separator.	read.csv2(file="myfile", header=TRUE)
read.delim()	Useful for reading delimited files, with tabs as the default separator.	read.delim(file="myfile", header=TRUE)
scan()	Allows you finer control over the read process when your data isn't tabular.	scan("myfile", skip = 1, nmax=100)
readLines()	Reads text from a text file one line at a time.	readLines("myfile")
read.fwf	Read a file with dates in fixed-width format. In other words, each column in the data has a fixed number of characters.	read.fwf("myfile", widths=c(1,2,3)

In addition to these options to read text data, the package foreign allows you to read data from other popular statistical formats, such as SPSS. To use these functions, you first have to load the built-in foreign package, with the following command:

```
> library("foreign")
```

The following table lists the functions to import data from SPSS, Stata, and SAS.

Function	What It Does	Example

read.spss	Reads SPSS data file	read.spss(“myfile”)
read.dta	Reads Stata binary file	read.dta(“myfile”)
read.xport	Reads SAS export file	read.export(“myfile”)