|  |  |
| --- | --- |
| **read.table {utils}** | R Documentation |

**Data Input**

**Description**

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.

**Usage**

**read.table**(file, header = FALSE, sep = "", quote = "\"'",

dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),

row.names, col.names, as.is = !stringsAsFactors,

na.strings = "NA", colClasses = NA, nrows = -1,

skip = 0, check.names = TRUE, fill = !blank.lines.skip,

strip.white = FALSE, blank.lines.skip = TRUE,

comment.char = "#",

allowEscapes = FALSE, flush = FALSE,

stringsAsFactors = default.stringsAsFactors(),

fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

**read.csv**(file, header = TRUE, sep = ",", quote = "\"",

dec = ".", fill = TRUE, comment.char = "", ...)

**read.csv2**(file, header = TRUE, sep = ";", quote = "\"",

dec = ",", fill = TRUE, comment.char = "", ...)

**read.delim**(file, header = TRUE, sep = "\t", quote = "\"",

dec = ".", fill = TRUE, comment.char = "", ...)

**read.delim2**(file, header = TRUE, sep = "\t", quote = "\"",

dec = ",", fill = TRUE, comment.char = "", ...)

**Arguments**

|  |  |
| --- | --- |
| file | the name of the file which the data are to be read from. Each row of the  table appears as one line of the file. If it does not contain an *absolute* path,  the file name is *relative* to the current working directory, [getwd](http://127.0.0.1:14695/library/utils/help/getwd)().  Tilde-expansion is performed where supported. This can be a compressed  file (see [file](http://127.0.0.1:14695/library/utils/help/file)).  Alternatively, file can be a readable text-mode [connection](http://127.0.0.1:14695/library/utils/help/connection) (which will be  opened for reading if necessary, and if so [close](http://127.0.0.1:14695/library/utils/help/close)d (and hence destroyed)  at the end of the function call). (If [stdin](http://127.0.0.1:14695/library/utils/help/stdin)() is used, the prompts for lines may  be somewhat confusing. Terminate input with a blank line or an EOF signal,  Ctrl-D on Unix and Ctrl-Z on Windows. Any pushback on stdin() will be  cleared before return.)  file can also be a complete URL. (For the supported URL schemes, see  the ‘URLs’ section of the help for [url](http://127.0.0.1:14695/library/utils/help/url).) |
| header | a logical value indicating whether the file contains the names of the  variables as its first line. If missing, the value is determined from the file  format: header is set to TRUE if and only if the first row contains one fewer  field than the number of columns. |
| sep | the field separator character. Values on each line of the file are separated  by this character. If sep = "" (the default for read.table) the separator is  ‘white space’, that is one or more spaces, tabs, newlines or carriage returns. |
| quote | the set of quoting characters. To disable quoting altogether, use  quote = "". See [scan](http://127.0.0.1:14695/library/utils/help/scan) for the behaviour on quotes embedded in quotes.  Quoting is only considered for columns read as character, which is all of  them unless colClasses is specified. |
| dec | the character used in the file for decimal points. |
| numerals | string indicating how to convert numbers whose conversion to double  precision would lose accuracy, see [type.convert](http://127.0.0.1:14695/library/utils/help/type.convert). Can be abbreviated.  (Applies also to complex-number inputs.) |
| row.names | a vector of row names. This can be a vector giving the actual row names,  or a single number giving the column of the table which contains the row  names, or character string giving the name of the table column containing  the row names.  If there is a header and the first row contains one fewer field than the  number of columns, the first column in the input is used for the row names.  Otherwise if row.names is missing, the rows are numbered.  Using row.names = NULL forces row numbering. Missing or NULL  row.names generate row names that are considered to be ‘automatic’  (and not preserved by [as.matrix](http://127.0.0.1:14695/library/utils/help/as.matrix)). |
| col.names | a vector of optional names for the variables. The default is to use  "V" followed by the column number. |
| as.is | the default behavior of read.table is to convert character variables  (which are not converted to logical, numeric or complex) to factors.  The variable as.is controls the conversion of columns not otherwise  specified by colClasses. Its value is either a vector of logicals (values  are recycled if necessary), or a vector of numeric or character indices  which specify which columns should not be converted to factors.  Note: to suppress all conversions including those of numeric columns,  set colClasses = "character".  Note that as.is is specified per column (not per variable) and so includes  the column of row names (if any) and any columns to be skipped. |
| na.strings | a character vector of strings which are to be interpreted as [NA](http://127.0.0.1:14695/library/utils/help/NA) values.  Blank fields are also considered to be missing values in logical, integer,  numeric and complex fields. Note that the test happens *after* white space  is stripped from the input, so na.strings values may need their own white  space stripped in advance. |
| colClasses | character. A vector of classes to be assumed for the columns. If unnamed,  recycled as necessary. If named, names are matched with unspecified  values being taken to be NA.  Possible values are NA (the default, when [type.convert](http://127.0.0.1:14695/library/utils/help/type.convert) is used), "NULL"  (when the column is skipped), one of the atomic vector classes (logical,  integer, numeric, complex, character, raw), or "factor", "Date" or "POSIXct".  Otherwise there needs to be an as method (from package **methods**) for  conversion from "character" to the specified formal class.  Note that colClasses is specified per column (not per variable) and so  includes the column of row names (if any). |
| nrows | integer: the maximum number of rows to read in. Negative and other invalid  values are ignored. |
| skip | integer: the number of lines of the data file to skip before beginning to read  data. |
| check.names | logical. If TRUE then the names of the variables in the data frame are  checked to ensure that they are syntactically valid variable names. If  necessary they are adjusted (by [make.names](http://127.0.0.1:14695/library/utils/help/make.names)) so that they are, and also  to ensure that there are no duplicates. |
| fill | logical. If TRUE then in case the rows have unequal length, blank fields  are implicitly added. See ‘Details’. |
| strip.white | logical. Used only when sep has been specified, and allows the stripping  of leading and trailing white space from unquoted character fields (numeric  fields are always stripped). See [scan](http://127.0.0.1:14695/library/utils/help/scan) for further details (including the exact  meaning of ‘white space’), remembering that the columns may include the  row names. |
| blank.lines.skip | logical: if TRUE blank lines in the input are ignored. |
| comment.char | character: a character vector of length one containing a single character or  an empty string. Use "" to turn off the interpretation of comments altogether. |
| allowEscapes | logical. Should C-style escapes such as \n be processed or read verbatim  (the default)? Note that if not within quotes these could be interpreted as a  delimiter (but not as a comment character). For more details see [scan](http://127.0.0.1:14695/library/utils/help/scan). |
| flush | logical: if TRUE, scan will flush to the end of the line after reading the last  of the fields requested. This allows putting comments after the last field. |
| stringsAsFactors | logical: should character vectors be converted to factors? Note that this is  overridden by as.is and colClasses, both of which allow finer control. |
| fileEncoding | character string: if non-empty declares the encoding used on a file (not a  connection) so the character data can be re-encoded. See the ‘Encoding’  section of the help for [file](http://127.0.0.1:14695/library/utils/help/file), the ‘R Data Import/Export Manual’ and ‘Note’. |
| encoding | encoding to be assumed for input strings. It is used to mark character  strings as known to be in Latin-1 or UTF-8 (see [Encoding](http://127.0.0.1:14695/library/utils/help/Encoding)): it is not used  to re-encode the input, but allows **R** to handle encoded strings in their  native encoding (if one of those two). See ‘Value’ and ‘Note’. |
| text | character string: if file is not supplied and this is, then data are read from  the value of text via a text connection. Notice that a literal string can be  used to include (small) data sets within R code. |
| skipNul | logical: should nuls be skipped? |
| ... | Further arguments to be passed to read.table. |

**Details**

This function is the principal means of reading tabular data into **R**.

Unless colClasses is specified, all columns are read as character columns and then converted using [type.convert](http://127.0.0.1:14695/library/utils/help/type.convert) to logical, integer, numeric, complex or (depending on as.is) factor as appropriate. Quotes are (by default) interpreted in all fields, so a column of values like "42" will result in an integer column.

A field or line is ‘blank’ if it contains nothing (except whitespace if no separator is specified) before a comment character or the end of the field or line.

If row.names is not specified and the header line has one less entry than the number of columns, the first column is taken to be the row names. This allows data frames to be read in from the format in which they are printed. If row.names is specified and does not refer to the first column, that column is discarded from such files.

The number of data columns is determined by looking at the first five lines of input (or the whole input if it has less than five lines), or from the length of col.names if it is specified and is longer. This could conceivably be wrong if fill or blank.lines.skip are true, so specify col.names if necessary (as in the ‘Examples’).

read.csv and read.csv2 are identical to read.table except for the defaults. They are intended for reading ‘comma separated value’ files (‘.csv’) or (read.csv2) the variant used in countries that use a comma as decimal point and a semicolon as field separator. Similarly, read.delim and read.delim2 are for reading delimited files, defaulting to the TAB character for the delimiter. Notice that header = TRUE and fill = TRUE in these variants, and that the comment character is disabled.

The rest of the line after a comment character is skipped; quotes are not processed in comments. Complete comment lines are allowed provided blank.lines.skip = TRUE; however, comment lines prior to the header must have the comment character in the first non-blank column.

Quoted fields with embedded newlines are supported except after a comment character. Embedded nuls are unsupported: skipping them (with skipNul = TRUE) may work.

**Value**

A data frame ([data.frame](http://127.0.0.1:14695/library/utils/help/data.frame)) containing a representation of the data in the file.

Empty input is an error unless col.names is specified, when a 0-row data frame is returned: similarly giving just a header line if header = TRUE results in a 0-row data frame. Note that in either case the columns will be logical unless colClasses was supplied.

Character strings in the result (including factor levels) will have a declared encoding if encoding is "latin1" or "UTF-8".

**Memory usage**

These functions can use a surprising amount of memory when reading large files. There is extensive discussion in the ‘R Data Import/Export’ manual, supplementing the notes here.

Less memory will be used if colClasses is specified as one of the six [atomic](http://127.0.0.1:14695/library/utils/help/atomic) vector classes. This can be particularly so when reading a column that takes many distinct numeric values, as storing each distinct value as a character string can take up to 14 times as much memory as storing it as an integer.

Using nrows, even as a mild over-estimate, will help memory usage.

Using comment.char = "" will be appreciably faster than the read.table default.

read.table is not the right tool for reading large matrices, especially those with many columns: it is designed to read *data frames* which may have columns of very different classes. Use [scan](http://127.0.0.1:14695/library/utils/help/scan) instead for matrices.

**Note**

The columns referred to in as.is and colClasses include the column of row names (if any).

There are two approaches for reading input that is not in the local encoding. If the input is known to be UTF-8 or Latin1, use the encoding argument to declare that. If the input is in some other encoding, then it may be translated on input. The fileEncoding argument achieves this by setting up a connection to do the re-encoding into the current locale. Note that on Windows or other systems not running in a UTF-8 locale, this may not be possible.

**References**

Chambers, J. M. (1992) *Data for models.* Chapter 3 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

**See Also**

The ‘R Data Import/Export’ manual.

[scan](http://127.0.0.1:14695/library/utils/help/scan), [type.convert](http://127.0.0.1:14695/library/utils/help/type.convert), [read.fwf](http://127.0.0.1:14695/library/utils/help/read.fwf) for reading *f*ixed *w*idth *f*ormatted input; [write.table](http://127.0.0.1:14695/library/utils/help/write.table); [data.frame](http://127.0.0.1:14695/library/utils/help/data.frame).

[count.fields](http://127.0.0.1:14695/library/utils/help/count.fields) can be useful to determine problems with reading files which result in reports of incorrect record lengths (see the ‘Examples’ below).

<https://tools.ietf.org/html/rfc4180> for the IANA definition of CSV files (which requires comma as separator and CRLF line endings).

**Examples**

## using count.fields to handle unknown maximum number of fields

## when fill = TRUE

test1 <- c(1:5, "6,7", "8,9,10")

tf <- tempfile()

writeLines(test1, tf)

read.csv(tf, fill = TRUE) # 1 column

ncol <- max(count.fields(tf, sep = ","))

read.csv(tf, fill = TRUE, header = FALSE,

col.names = paste0("V", seq\_len(ncol)))

unlink(tf)

## "Inline" data set, using text=

## Notice that leading and trailing empty lines are auto-trimmed

read.table(header = TRUE, text = "

a b

1 2

3 4

")