

# Package: csv (via r-universe)

October 9, 2024

**Type** Package

**Title** Read and Write CSV Files with Selected Conventions

**Version** 0.6.2

**Author** Tim Bergsma

**Maintainer** Tim Bergsma <bergsmat@gmail.com>

**Description** Reads and writes CSV with selected conventions. Uses the same generic function for reading and writing to promote consistent formats.

**License** GPL-3

**Imports** data.table, stringi

**RoxygenNote** 7.1.1

**Repository** <https://bergsmat.r-universe.dev>

**RemoteUrl** <https://github.com/bergsmat/csv>

**RemoteRef** HEAD

**RemoteSha** 819597d0bfe1612da78825e8299eb47c1c0ca39a

## Contents

as.csv . . . . .	2
as.csv.character . . . . .	2
as.csv.data.frame . . . . .	3

<b>Index</b>	<b>5</b>
--------------	----------

---

`as.csv`*Read or Write CSV Using Selected Conventions*

---

### Description

Reads or writes CSV files in a conventional way. Generic, with methods for `character` and `data.frame`. A length-one `character` argument is treated as a filepath and tries to return a `data.frame`. A `data.frame` argument is written to the specified filepath. Typically, `quote` and `row.names` are `FALSE` and `na` is `"."`. When reading, white space and empty strings are treated as `NA`, and `strip.white` is `TRUE`. When writing, values with commas or double-quotes are double-quoted (and embedded double-quotes are doubled).

### Usage

```
as.csv(x, ...)
```

### Arguments

<code>x</code>	object
<code>...</code>	passed arguments

### See Also

[as.csv.character](#), [as.csv.data.frame](#)

Other `as.csv`: [as.csv.character\(\)](#), [as.csv.data.frame\(\)](#)

### Examples

```
data <- head(Theoph)
filepath <- file.path(tempdir(), 'theoph.csv')
as.csv(data, filepath)
as.csv(filepath)
```

---

`as.csv.character`*Treat Character as CSV filename.*

---

### Description

Treat a character string as a CSV filename.

**Usage**

```
## S3 method for class 'character'
as.csv(
  x,
  as.is = TRUE,
  na.strings = c("", "\\s", ".", "NA"),
  strip.white = TRUE,
  check.names = FALSE,
  source = getOption("csv_source", TRUE),
  ...
)
```

**Arguments**

x	character file path
as.is	passed to <a href="#">read.csv</a>
na.strings	passed to <a href="#">read.csv</a>
strip.white	passed to <a href="#">read.csv</a>
check.names	passed to <a href="#">read.csv</a>
source	whether to assign x as the source attribute of the return value
...	passed to <a href="#">read.csv</a> if accepted by <a href="#">read.table</a>

**Details**

If x is character, is length one, and is a path to a file, an attempt is made to read the file.

**Value**

data.frame, with attribute 'source' set to x

**See Also**

Other as.csv: [as.csv.data.frame\(\)](#), [as.csv\(\)](#)

---

as.csv.data.frame      *Save a Data Frame as CSV.*

---

**Description**

Saves a data.frame as CSV, using selected conventions.

**Usage**

```
## S3 method for class 'data.frame'
as.csv(x, file, na = ".", quote = FALSE, auto = !quote, row.names = FALSE, ...)
```

**Arguments**

x	data.frame
file	passed to <a href="#">write.csv</a>
na	passed to <a href="#">write.csv</a>
quote	passed to <a href="#">write.csv</a>
auto	double-quote column names and row values with embedded commas or double-quotes; the latter are escaped by doubling them
row.names	passed to <a href="#">write.csv</a>
...	passed to <a href="#">write.csv</a> if accepted by <a href="#">write.table</a>

**Value**

invisible data.frame (x)

**See Also**

Other as.csv: [as.csv.character\(\)](#), [as.csv\(\)](#)

**Examples**

```
x <- data.frame(
  check.names = FALSE,
  stringsAsFactors = TRUE,
  person = 1:3,
  `name, suffix` = c("Bill Smith", "Joseph "Joe" Hancock", "Mary Laguire, DDS")
)
file <- tempfile()
as.csv(x, file)
y <- as.csv(file, as.is=FALSE)
attr(y, 'source')
attr(y, 'source') <- NULL
x
y
stopifnot(identical(x,y))
```

# Index

## \* **as.csv**

as.csv, [2](#)

as.csv.character, [2](#)

as.csv.data.frame, [3](#)

as.csv, [2](#), [3](#), [4](#)

as.csv.character, [2](#), [2](#), [4](#)

as.csv.data.frame, [2](#), [3](#), [3](#)

read.csv, [3](#)

read.table, [3](#)

write.csv, [4](#)

write.table, [4](#)