# Package ‘orgutils’

December 15, 2025

**Type** Package

**Title** Helper Functions for Org Files

**Version** 0.5-2

**Date** 2025-12-15

**Maintainer** Enrico Schumann <es@enricoschumann.net>

**Description** Helper functions for Org files (<https://orgmode.org/>): a generic function 'toOrg' for transforming R objects into Org markup (most useful for data frames; there are also methods for Dates/POSIXt) and a function to read Org tables into data frames.

**License** GPL (>= 2)

**Depends** R (>= 3.2)

**Suggests** tinytest

**URL** https://enricoschumann.net/R/packages/orgutils/ , https://sr.ht/~enricoschumann/orgutils/

**NeedsCompilation** no

**Author** Enrico Schumann [aut, cre] (ORCID: <https://orcid.org/0000-0001-7601-6576>)

**Repository** CRAN

**Date/Publication** 2025-12-15 16:30:10 UTC

## Contents

## orgutils-package *Org Utils*

### Description

Helper functions to interact with Org files: read Org tables, convert R objects to Org markup.

### Details

Org mode is a major mode for Emacs; see `https://orgmode.org/manual/Summary.html#Summary` for a summary of what it does.

The **orgutils** package provides helper functions for interacting with Org files (reading Org tables, convert R objects to Org markup) without Emacs. Since Org syntax is very human-readable, such conversions are useful also, for instance, in plain-text emails or reports.

There are several other packages that help you work with Org files as well, such as **orgR** or **ascii**.

### Author(s)

Enrico Schumann <es@enricoschumann.net>

### References

Org mode manual `https://orgmode.org/`

### See Also

`toOrg`, `readOrg`

## readOrg *Read Org Tables*

### Description

Read an Org table from a file.

### Usage

```
readOrg(file, header = TRUE, dec = ".", comment.char = "",
        encoding = "", strip.white = TRUE,
        stringsAsFactors = FALSE,
        table.name = NULL, text,
        table.missing = NULL, ...,
        strip.format = TRUE,
        strip.horiz.rules = TRUE,
        collapse.header = FALSE)
```

## Arguments

| | |
|---|---|
| `file` | character |
| `header` | logical: If `TRUE`, and `collapse.header` is `FALSE`, the first row of the table is used for column names (`strip.horiz.rules` determines whether initial rules are removed first). |
| `dec` | character; see `read.table` |
| `comment.char` | character; see `read.table` |
| `encoding` | string; see `read.table` |
| `strip.white` | logical; see `read.table` |
| `strip.format` | logical: strip rows of format instructions, such as <c> |
| `strip.horiz.rules` | logical: string horizontal rules from table |
| `collapse.header` | logical: if `TRUE`, all rows before the first horizontal rule are considered table headers (as defined in the Org manual) |
| `stringsAsFactors` | logical: note that the default `FALSE` differs from read.csv |
| `table.name` | character: a regex; the name of the table to read. |
| `text` | character: if `file` is not supplied, `text` is read via `textConnection` |
| `table.missing` | what to do if a table specified by `table.name` is not found. Default is to return `NULL`. Set to string `"stop"` to throw an error. |
| `...` | further arguments |

## Details

Org tables are very human-readable plain-text tables that look like

```
| Column1 | Column2 |
|---------+---------|
|       1 |       2 |
|       3 |       4 |
```

A line that starts with '`|`' (after optional whitespace) is considered a table row; a line that starts with '`|-`' (after optional whitespace) is a horizontal rule. Rows before the first horizontal rule are header lines (see the Org manual).

Depending on the settings of `strip.format` and `strip.horiz.rules`, format instructions such as <5> and are discarded. Then the function uses `read.csv` to read the remainder of the file/table.

When `table.name` is specified, the function looks for a line that starts with `#+NAME: <table.name>` and reads the table that follows that line.

For empty files, `readOrg` behaves like read.csv: when completely empty, it fails; when headers are found, a zero-row data.frame is returned.

## Value

a `data.frame`

## Author(s)

Enrico Schumann

## References

Org manual https://orgmode.org/manual/Tables.html

## See Also

read.csv

## Examples

```
## create an Org file with a table and read the table
tmp <-
"#+TITLE: A Table

Next comes a table.

#+name: test_table
| a | b |
|---+---|
| 1 | 2 |
| 3 | 4 |

That was a table.
"

fname <- tempfile("testfile", fileext = ".org")
writeLines(tmp, fname)

library("orgutils")
readOrg(fname, table.name = "test_table")
```

---

## toOrg *Generate Org-mode Markup*

---

## Description

Transform R objects into Org-mode objects.

## Usage

```
toOrg(x, ...)

## S3 method for class 'org'
print(x, ...)
```

```
## S3 method for class 'data.frame'
toOrg(x, row.names = NULL, ...)

## S3 method for class 'Date'
toOrg(x, inactive = FALSE, ...)

## S3 method for class 'POSIXt'
toOrg(x, inactive = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | an object |
| `row.names` | `NULL`, logical or character. If `TRUE`, `row.names` of `x` are added as the first column, with column name `"row.names"`. If a character string, the string is used as the column name. See Examples. If `NULL`, `row.names` are added when they are not `1, 2, ...` (i.e. row numbers). If `FALSE`, `row.names` are not added. |
| `inactive` | logical: use inactive timestamps? See `https://orgmode.org/manual/Creating-timestamps.html` . |
| `...` | other arguments |

## Details

Transforms an object `x` into character vectors with Org markup. Most useful when `x` is a data.frame.

`toOrg` is meant for snippets of code, not for producing whole Org documents.

When you work with `POSIXt`, make sure that a potential timezone does not cause trouble: Org does not support timezones.

## Value

A character vector, usually with class `org`. In some cases, class `character` is additionally attached.

To save it to a file, use `writeLines`.

## Author(s)

Enrico Schumann

## References

Org mode manual `https://orgmode.org/manual/index.html`

## See Also

`toLatex`, function `as.orgtable` in **microplot**

## Examples

```
toOrg(data.frame(a = 1:3, row.names = LETTERS[1:3]))
## =>  | row.names | a |
##     |-----------+---|
##     | A         | 1 |
##     | B         | 2 |
##     | C         | 3 |

toOrg(data.frame(a = 1:3))
## =>  | a |
##     |---|
##     | 1 |
##     | 2 |
##     | 3 |

toOrg(data.frame(a = 1:3), row.names = TRUE)
## =>  | row.names | a |
##     |-----------+---|
##     | 1         | 1 |
##     | 2         | 2 |
##     | 3         | 3 |


toOrg(data.frame(a = 1:5), row.names = "row numbers")
## =>  | row numbers | a |
##     |-------------+---|
##     | 1           | 1 |
##     | 2           | 2 |
##     | 3           | 3 |
##     | 4           | 4 |
##     | 5           | 5 |

## Not run:
writeLines(toOrg(data.frame(a = 1:3)), "~/Desktop/my_table.org")
## End(Not run)

## Dates/Times
toOrg(as.Date("2015-01-01"))                   ## <2015-01-01 Thu>
toOrg(as.Date("2015-01-01"), inactive = TRUE)  ## [2015-01-01 Thu]
toOrg(Sys.time())                              ## <2017-03-20 Mon 13:23:18>

## Convert Org dates to Date

## see ?strptime: Each input string is processed as far as
##                necessary for the format specified: any
##                trailing characters are ignored.
d <- toOrg(as.Date("2015-01-01"))
as.Date(d, "<%Y-%m-%d")
```

# Index