# Package: zuxlsx (via r-universe)

November 15, 2026

**Title** Read XLSX Workbooks Without System Dependencies

**Version** 0.1.0

**Description** Reads spreadsheet data from XLSX workbooks using a bundled copy of the 'xlsxio' reader (<https://github.com/brechtsanders/xlsxio>), so that no system XML or ZIP library is required. The 'Expat' parser, the 'miniz' ZIP reader and the 'TF-PSA-Crypto' cryptography used to open password-protected workbooks are linked statically at install time from the 'zuxml', 'zukomp' and 'zucrypt' packages, which means there is no run-time dependency on any of them. Worksheet XML is parsed as a stream rather than loaded whole; the shared-string table and the cells being returned are held, and a password-protected workbook is decrypted into memory first. Writing workbooks is out of scope.

**License** MIT + file LICENSE

**Copyright** See inst/COPYRIGHTS and tools/vendor/manifest.tsv.

**URL** https://github.com/pedrobtz/zuxlsx, https://pedrobtz.github.io/zuxlsx/

**BugReports** https://github.com/pedrobtz/zuxlsx/issues

**Depends** R (>= 4.1)

**Encoding** UTF-8

**Language** en-GB

**Roxygen** list(markdown = TRUE)

**LinkingTo** zucrypt, zukomp, zuxml

**Suggests** testthat (>= 3.0.0), withr, zucrypt

**Config/testthat/edition** 3

**Config/Needs/website** knitr, rmarkdown

**Config/roxygen2/version** 8.1.0

**Repository** https://pedrobtz.r-universe.dev

**Date/Publication** 2026-11-15 06:29:23 UTC

**RemoteUrl** https://github.com/pedrobtz/zuxlsx

**RemoteRef** HEAD

**RemoteSha** ffcb83de9ce9ea9784057e3a3b15819811cc1c97

# Contents

xlsx_cells . . . . . 2
xlsx_info . . . . . 3
xlsx_read . . . . . 4
xlsx_rows . . . . . 5
xlsx_sheets . . . . . 6
xlsx_stream_cells . . . . . 7
zuxlsx-conditions . . . . . 9

**Index** **11**

---

`xlsx_cells` *Read the cells of a worksheet*

---

## Description

Reads a worksheet one cell at a time and returns them in a data frame, one row per cell. This is the low-level reader: it reports what is in each cell without assembling columns or guessing a type for them; `xlsx_read()` does that.

## Usage

```
xlsx_cells(path, sheet = 1, password = NULL)
```

## Arguments

| | |
|---|---|
| `path` | Path to an `.xlsx` file. |
| `sheet` | The worksheet to read: either its name, or its position in the workbook.<br>Cells the worksheet leaves out between the ones it writes come back as `"blank"`, and so may a few the file never names: the reader pads to keep positions aligned. `xlsx_read()` ignores blank cells when it decides where a table starts and ends. |
| `password` | The password of a password-protected workbook, or `NULL` (the default) for one without. The workbook is decrypted in memory and never written to disk. Agile encryption – what Excel has written since 2013 – is supported; the older standard encryption is not, and is reported as such. Ignored for a workbook that is not encrypted. R cannot erase a string, so the password may stay in memory after the call. |

## Value

A data frame with one row per cell and the columns:

`row`, `col` Position, counting from 1.

`type` One of `"blank"`, `"number"`, `"string"`, `"boolean"`, `"error"` or `"date"`.

`value` The cell as written, as a string.

`number` The numeric value for `"number"`, `"boolean"` and `"date"` cells, and `NA` otherwise. A date is its Excel serial number; it is not converted here, because the epoch is a property of the workbook rather than of the cell. A date written as ISO 8601 text (`t="d"`) is given the serial an Excel number would have; one that cannot be read as a date is a `"string"`.

## See Also

`xlsx_sheets()` to list the worksheets, and zuxlsx-conditions for the errors this can raise.

## Examples

```
path <- system.file("extdata", "two-sheets.xlsx", package = "zuxlsx")
cells <- xlsx_cells(path)
head(cells)

# The type is the cell's own, before any column is built from it. A date is
# reported as a date and its serial number given unconverted, because the
# epoch belongs to the workbook rather than to the cell.
table(cells$type)
attr(cells, "date1904")

# The second worksheet, by name rather than by position.
xlsx_cells(path, "notes")
```

---

# xlsx_info *Report the native libraries zuxlsx was built against*

---

## Description

zuxlsx vendors the 'xlsxio' reader and links 'Expat', 'miniz' and 'TF-PSA-Crypto' statically out of the `zuxml`, `zukomp` and `zucrypt` packages, through `LinkingTo`. This reports what it actually got, which is the quickest way to tell a stale build from a current one.

## Usage

```
xlsx_info()
```

## Value

A list with elements `xlsxio`, `expat`, `miniz` and `tf_psa_crypto`. The last is `NA` if the cryptographic backend failed to start when the package loaded.

## Examples

```
xlsx_info()
```

---

`xlsx_read` *Read a worksheet into a data frame*

---

## Description

Reads a worksheet and assembles its cells into columns, giving each column the type its cells support. This is the high-level reader; `xlsx_cells()` gives the same worksheet cell by cell.

## Usage

```
xlsx_read(path, sheet = 1, col_names = TRUE, range = NULL, password = NULL)
```

## Arguments

| | |
|---|---|
| `path` | Path to an `.xlsx` file. |
| `sheet` | The worksheet to read: either its name, or its position in the workbook.<br>Cells the worksheet leaves out between the ones it writes come back as `"blank"`, and so may a few the file never names: the reader pads to keep positions aligned. `xlsx_read()` ignores blank cells when it decides where a table starts and ends. |
| `col_names` | Whether the first row holds column names. When `FALSE`, columns are named `X1`, `X2` and so on. When `TRUE`, a column whose header cell is blank is named `X` and its position, and repeated names are made unique with `make.unique()` and an underscore, so a second `id` becomes `id_1`. |
| `range` | An A1-style cell range limiting what is read, or `NULL` for the whole worksheet. Either corner may name a cell, a column or a row, so `"B2:D10"` takes a rectangle, `"A:C"` takes three columns in full, and `"2:10"` takes nine rows in full. The corners may be given in either order. A range is read as its own rectangle: cells outside it are ignored, the top-left becomes the first row and column, and columns the range covers appear even where the worksheet left them empty. When `col_names` is `TRUE` the first row of the range supplies the names. A range may not reach past `XFD1048576`, the last cell a worksheet can have. |
| `password` | The password of a password-protected workbook, or `NULL` (the default) for one without. The workbook is decrypted in memory and never written to disk. Agile encryption – what Excel has written since 2013 – is supported; the older standard encryption is not, and is reported as such. Ignored for a workbook that is not encrypted. R cannot erase a string, so the password may stay in memory after the call. |

## Details

Without `range`, the table is the rectangle its non-blank cells span: empty rows above and below it, and empty columns either side, are not part of it, so a table that starts at `B3` reads with its header from row 3. Inside that rectangle a worksheet may leave a row out of its XML rather than write an empty one; such a row is kept, as a row of `NA`, wherever it falls – including directly beneath the header.

Column types are inferred by promotion. A column of blanks is logical, one of booleans stays logical, numbers give a double, and anything that mixes types, or that holds a string or a cell error, becomes character. A column whose cells are all dates becomes a `Date`, or a `POSIXct` when any of them carries a time of day.

## Value

A data frame with one column per worksheet column, typed as described above. A worksheet, or a range, with no cells in it gives a data frame with no columns and no rows.

## See Also

`xlsx_cells()` for the cells themselves, and `xlsx_sheets()` to list the worksheets.

## Examples

```
path <- system.file("extdata", "two-sheets.xlsx", package = "zuxlsx")

# Each column takes the type its cells support.
readings <- xlsx_read(path)
readings
vapply(readings, function(x) class(x)[1], character(1))

# A worksheet by name, and part of one by range.
xlsx_read(path, "notes")
xlsx_read(path, range = "B1:C3")

# Without a header row, columns are named by position.
xlsx_read(path, range = "A2:B4", col_names = FALSE)
```

---

# xlsx_rows *Read a worksheet as rows*

---

## Description

Returns a worksheet row by row, as the text of each cell, without inferring a type for any column. This is the row-oriented counterpart to `xlsx_cells()`, and is the shape to reach for when a worksheet is not rectangular enough for `xlsx_read()` – a report with stacked blocks, or a file whose header is several rows deep.

## Usage

```
xlsx_rows(path, sheet = 1, password = NULL)
```

### Arguments

| | |
|---|---|
| path | Path to an .xlsx file. |
| sheet | The worksheet to read: either its name, or its position in the workbook.<br>Cells the worksheet leaves out between the ones it writes come back as "blank", and so may a few the file never names: the reader pads to keep positions aligned. xlsx_read() ignores blank cells when it decides where a table starts and ends. |
| password | The password of a password-protected workbook, or NULL (the default) for one without. The workbook is decrypted in memory and never written to disk. Agile encryption – what Excel has written since 2013 – is supported; the older standard encryption is not, and is reported as such. Ignored for a workbook that is not encrypted. R cannot erase a string, so the password may stay in memory after the call. |

### Value

A list with one character vector per row, from row 1 of the worksheet to the last row holding a cell, so the nth element is row n. Every vector is as long as the widest row, so the nth element of each is column n. Blank cells, and rows with none, are NA.

### See Also

xlsx_cells() for typed cells, xlsx_read() for a data frame.

### Examples

```
path <- system.file("extdata", "two-sheets.xlsx", package = "zuxlsx")
rows <- xlsx_rows(path)

# One character vector per row, every one as wide as the widest, with no
# type inference: a number is its text.
rows[[1]]
rows[[2]]
```

---

## xlsx_sheets *List the worksheets in an xlsx workbook*

---

### Description

List the worksheets in an xlsx workbook

### Usage

```
xlsx_sheets(path, password = NULL)
```

### Arguments

| | |
|---|---|
| path | Path to an .xlsx file. |
| password | The password of a password-protected workbook, or NULL (the default) for one without. The workbook is decrypted in memory and never written to disk. Agile encryption – what Excel has written since 2013 – is supported; the older standard encryption is not, and is reported as such. Ignored for a workbook that is not encrypted. R cannot erase a string, so the password may stay in memory after the call. |

### Value

A character vector of worksheet names, in workbook order. A workbook always has at least one.

### See Also

zuxlsx-conditions for the errors this can raise.

### Examples

```
path <- system.file("extdata", "two-sheets.xlsx", package = "zuxlsx")
xlsx_sheets(path)
```

---

## xlsx_stream_cells *Read a worksheet in chunks*

---

### Description

Reads a worksheet a piece at a time, passing each piece to a function rather than returning the whole thing. This is the low-level streaming reader: use it when a worksheet is too large to hold, or when the answer can be found without reading all of it.

### Usage

```
xlsx_stream_cells(
  path,
  sheet = 1,
  callback,
  chunk_size = 10000L,
  password = NULL
)
```

### Arguments

| | |
|---|---|
| path | Path to an .xlsx file. |
| sheet | The worksheet to read: either its name, or its position in the workbook. Cells the worksheet leaves out between the ones it writes come back as "blank", and so may a few the file never names: the reader pads to keep positions aligned. xlsx_read() ignores blank cells when it decides where a table starts and ends. |

| | |
|---|---|
| callback | A function of one argument, called with each chunk of cells. Return FALSE to stop reading. |
| chunk_size | The least number of cells to gather before calling callback. |
| password | The password of a password-protected workbook, or NULL (the default) for one without. The workbook is decrypted in memory and never written to disk. Agile encryption – what Excel has written since 2013 – is supported; the older standard encryption is not, and is reported as such. Ignored for a workbook that is not encrypted. R cannot erase a string, so the password may stay in memory after the call. |

## Details

callback is called with a data frame of cells in the shape xlsx_cells() returns. Returning FALSE from it stops the read, leaving the rest of the worksheet unparsed; any other value continues.

A chunk never splits a row, so a callback can rely on seeing every cell of a row together. chunk_size is therefore a lower bound rather than an exact count: a chunk ends at the first row boundary at or after it.

## Value

TRUE if callback stopped the read, FALSE if the worksheet was read to the end. Returned invisibly.

## See Also

xlsx_cells(), which reads a whole worksheet at once, and xlsx_read(), which builds columns from it.

## Examples

```
path <- system.file("extdata", "two-sheets.xlsx", package = "zuxlsx")

# Count the cells of each type without holding the worksheet.
totals <- 0
xlsx_stream_cells(path, 1, function(cells) {
  totals <<- totals + table(cells$type)
})
totals

# Stop as soon as something is found. The rest of the sheet is not parsed.
first <- NULL
xlsx_stream_cells(path, 1, function(cells) {
  hit <- cells[cells$type == "date", ]
  if (nrow(hit) > 0) {
    first <<- hit[1, ]
    FALSE
  }
})
first
```

`zuxlsx-conditions` *Conditions raised by zuxlsx*

## Description

All errors raised by zuxlsx carry a condition class, so that they can be caught by kind rather than by matching on the message text, which is not stable. Every class below is a subclass of `zuxlsx_error`.

## Details

`zuxlsx_input_error` An argument was not usable: a `path` that is not a single string, does not exist or is a directory; a `sheet`, `range`, `col_names`, `callback`, `chunk_size` or `password` of the wrong kind; or a `range` reaching past `XFD1048576`.

`zuxlsx_zip_error` The file could not be opened as a ZIP archive – it is unreadable, truncated, or not a ZIP at all – or a worksheet's compressed data failed its integrity check.

`zuxlsx_xml_error` A part of the workbook is not well-formed XML. Raised in preference to `zuxlsx_ooxml_error` when the failure is the XML itself rather than what it says, and names the part or worksheet and the line. A worksheet that breaks off part-way is reported this way rather than read as a shorter one.

`zuxlsx_ooxml_error` The XML is well formed but is not a valid workbook: the archive declares no worksheet, or a worksheet refers to a cell beyond `XFD1048576` or to a shared string that does not exist.

`zuxlsx_sheet_error` The workbook opened, but the requested worksheet is not in it.

`zuxlsx_encrypted_error` The workbook is password-protected and no `password` was given. It is also a subclass of `zuxlsx_unsupported_format_error`, so a handler for that class catches it too, while one for `zuxlsx_encrypted_error` alone can prompt for a password.

`zuxlsx_password_error` The `password` given is not the workbook's. A subclass of `zuxlsx_encrypted_error`, so one handler covers a password that is missing and one that is wrong. A damaged encryption header is indistinguishable from a wrong password, by design of the format.

`zuxlsx_integrity_error` A password-protected workbook is damaged: its encrypted package does not match the integrity code stored with it, or its container or encryption parameters are inconsistent. Nothing is decrypted from a file that fails this check.

`zuxlsx_unsupported_format_error` The file is a spreadsheet, but not one zuxlsx can read: an `.xlsb`, whose worksheets are binary rather than XML; a legacy `.xls`; or a workbook encrypted with a scheme other than agile encryption, such as Office 2007's standard encryption. Raised in preference to reporting such a file as corrupt, which is what it otherwise looks like.

`zuxlsx_memory_error` An allocation failed while reading.

A failure of the cryptographic backend while decrypting, and an internal error that should not happen, carry `zuxlsx_error` alone.

Conditions carry the offending `path` in a `path` element, where one applies.

## Examples

```
tryCatch(
  xlsx_sheets("no-such-file.xlsx"),
  zuxlsx_input_error = function(e) conditionMessage(e)
)
```

# Index

make.unique(), *4*

xlsx_cells, 2
xlsx_cells(), *4–6*, *8*
xlsx_info, 3
xlsx_read, 4
xlsx_read(), *2*, *4–8*
xlsx_rows, 5
xlsx_sheets, 6
xlsx_sheets(), *3*, *5*
xlsx_stream_cells, 7

zuxlsx-conditions, *3*, *7*, 9