Package 'unexcel'

July 28, 2026

Title Revert Spreadsheet Date Auto-Conversion to the Numbers Typed

Version 0.2.0

Description Spreadsheets silently turn entries such as '30.3' into dates, so the imported data carry date serials instead of the numbers that were typed. Reading the workbook directly recovers those numbers without guesswork: an 'xlsx' file states its own date system, and records which cells are formatted as dates and in which field order, so the values to repair are identified from the file rather than inferred from their magnitude. Functions are also provided for data already imported, where that evidence is no longer available, using conservative and configurable detection.

License MIT + file LICENSE

URL https://github.com/drhrf/unexcel, https://drhrf.github.io/unexcel/

BugReports https://github.com/drhrf/unexcel/issues

Depends R (>= 3.6)

Imports stats, utils, xml2 (>= 1.3.0)

Suggests knitr (>= 1.40), pkgdown (>= 2.0.0), rmarkdown (>= 2.20), testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

Config/testthat/edition 3

Config/Needs/website pkgdown

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Hercules R. Freitas [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-1584-9157>)

Maintainer Hercules R. Freitas <hercules.freitas@uerj.br>

Repository CRAN

Date/Publication 2026-07-28 15:30:02 UTC

Contents

excel_date_cells . . . 2
excel_date_columns . . . 3
excel_date_system . . . 4
excel_origin . . . 4
excel_sheets . . . 5
fix_serial_columns . . . 6
restore_day_month . . . 7
serial_to_day_month . . . 9
unexcel_xlsx . . . 10

Index **12**

`excel_date_cells` *Find the cells a workbook formats as dates*

Description

Reports every cell whose style resolves to a date number format, together with the serial it stores and the `day.month` value that serial restores to. This is the audit trail behind `unexcel_xlsx()`: nothing here is a guess, so the output can be checked cell by cell before any value is changed.

Usage

```
excel_date_cells(path, sheet = 1)
```

Arguments

`path`	Path to an `.xlsx` file.
`sheet`	Sheet name, or its index in the workbook. Defaults to the first sheet.

Details

A cell is reported when its number format is one of Excel's built-in date formats, or a custom format whose code contains date tokens, *and* it holds a number. Text that merely looks like a date is left alone.

Value

A data frame with one row per date-formatted cell and the columns `ref`, `row`, `col`, `serial`, `date`, `num_fmt_id`, `format_code`, `field_order` and `day_month`. The `date_system` attribute records the system the workbook declares. Zero rows means the workbook formats nothing as a date – there is nothing to restore.

See Also

`unexcel_xlsx()` to apply the restoration, `excel_date_columns()` to summarise this by column.

Examples

```
path <- system.file("extdata", "typed-numbers.xlsx", package = "unexcel")
excel_date_cells(path)
```

`excel_date_columns` *Summarise date-formatted cells by column*

Description

A column-level view of `excel_date_cells()`, for when a workbook has already been imported with another reader and only the *metadata* is needed. The `name` and `field_order` columns say which imported columns to repair and in which order to read the fields, so `restore_day_month()` can be applied to exactly those columns with nothing left to infer.

Usage

```
excel_date_columns(path, sheet = 1, col_names = TRUE)
```

Arguments

`path`	Path to an `.xlsx` file.
`sheet`	Sheet name, or its index in the workbook. Defaults to the first sheet.
`col_names`	If `TRUE` (default), the first row of the sheet holds column names, as it would for `utils::read.csv()` or `readxl::read_excel()`.

Value

A data frame with one row per column containing at least one date-formatted cell: `col` (index), `name`, `n_date` (date-formatted cells), `n_values` (populated cells below the header), `prop_date`, `field_order`, and `format_code`.

Examples

```
path <- system.file("extdata", "typed-numbers.xlsx", package = "unexcel")
cols <- excel_date_columns(path)
cols

# Repair a frame imported by any other reader, with no guessing left:
df <- unexcel_xlsx(path, restore = FALSE)
for (i in seq_len(nrow(cols))) {
  df[[cols$name[i]]] <- restore_day_month(
    df[[cols$name[i]]],
    date_system = excel_date_system(path),
    order = cols$field_order[i]
  )
}
```

excel_date_system *Read the date system a workbook declares*

Description

Excel stores dates as a day count, but from one of two origins. Which one is recorded in the workbook itself, in the `date1904` attribute of `xl/workbook.xml` – so it can be read rather than inferred. Passing the result to `serial_to_day_month()` or `restore_day_month()` removes the only assumption that can silently shift every restored value by four years and one day.

Usage

```
excel_date_system(path)
```

Arguments

`path`	Path to an `.xlsx` file.

Value

`"1900"` or `"1904"`.

See Also

`excel_origin()`, `excel_date_cells()`, `unexcel_xlsx()`.

Examples

```
path <- system.file("extdata", "typed-numbers.xlsx", package = "unexcel")
excel_date_system(path)
```

excel_origin *The origin date behind each Excel date system*

Description

The origin date behind each Excel date system

Usage

```
excel_origin(date_system = c("1900", "1904"))
```

Arguments

`date_system`	`"1900"` or `"1904"`.

Details

Excel's 1900 system counts 1900-01-01 as day 1 but also includes the non-existent 1900-02-29, so an origin of 1899-12-30 reproduces what Excel displays for every serial from 61 onward. The 1904 system, inherited from early Macintosh versions, counts 1904-01-01 as day 0.

Value

A length-one `Date`.

Examples

```
excel_origin("1900")
as.Date(45746, origin = excel_origin("1900"))
```

`excel_sheets` *List the sheets in a workbook*

Description

List the sheets in a workbook

Usage

```
excel_sheets(path)
```

Arguments

`path`	Path to an `.xlsx` file.

Value

A character vector of sheet names, in workbook order.

Examples

```
path <- system.file("extdata", "typed-numbers.xlsx", package = "unexcel")
excel_sheets(path)
```

`fix_serial_columns` *Revert Excel date serials across a data frame*

Description

Applies `restore_day_month()` to the columns that look dominated by Excel serials, leaving the rest of the frame alone. Column selection is a heuristic, controlled by `pmin`: it is the price of working from imported values. If the original workbook is available, `unexcel_xlsx()` selects columns from the file's own date formatting instead, and `excel_date_columns()` reports that selection for a frame imported by any other reader.

Usage

```
fix_serial_columns(
  df,
  pmin = 0.8,
  cols = NULL,
  low_serial = 20000,
  high_serial = 65000,
  year_window = 1990:2035,
  date_system = c("1900", "1904"),
  order = c("dm", "md"),
  output = c("numeric", "character"),
  origin_mode = NULL,
  ref_date = Sys.Date()
)
```

Arguments

`df`	A data frame.
`pmin`	Minimum fraction of a column's values that must be in-range whole numbers before the column is treated as serials.
`cols`	Optional column names or indices to restrict the operation to. Supplying this turns off the `pmin` scan, so nothing is selected by guesswork; pair it with `excel_date_columns()` for a fully determined result.
`low_serial, high_serial`	Bounds, inclusive, on what counts as a plausible serial. The defaults span roughly 1954 to 2078.
`year_window`	Years that a converted serial may resolve to. A serial landing outside the window is left alone.
`date_system`	`"1900"` or `"1904"`; see `excel_date_system()`.
`order`	`"dm"` or `"md"`; see `serial_to_day_month()`.
`output`	`"numeric"` or `"character"`; see `serial_to_day_month()`. `"character"` preserves a trailing zero, so a value typed as `3.10` survives as `"3.10"` rather than `3.1`.

origin_mode — Deprecated. "1900" and "1904" are passed to date_system. "auto" re-enables the 0.1.0 behaviour of picking the date system by comparing candidate dates against ref_date, with a warning: that comparison is a heuristic and can shift every restored value by four years and a day.

ref_date — Reference date used only by origin_mode = "auto".

Value

The data frame, with the selected columns restored.

See Also

unexcel_xlsx(), excel_date_columns().

Examples

```
df <- data.frame(a = c(45746, 45823), b = c(1.2, 3.4))
fix_serial_columns(df)

# Name the columns yourself and nothing is inferred
fix_serial_columns(df, cols = "a")
```

---

restore_day_month *Revert Excel date serials in an already-imported vector*

Description

Restores the day.month values a spreadsheet auto-converted to dates, working from the imported numbers alone. Use this when the original .xlsx is gone – a CSV export, a legacy .xls, a data frame handed over by a collaborator. When the workbook *is* available, use unexcel_xlsx() instead: it reads the date system and the date formatting out of the file, so it needs none of the guardrails below.

Usage

```
restore_day_month(
  x,
  low_serial = 20000,
  high_serial = 65000,
  year_window = 1990:2035,
  date_system = c("1900", "1904"),
  order = c("dm", "md"),
  output = c("numeric", "character"),
  origin_mode = NULL,
  ref_date = Sys.Date()
)
```

Arguments

x	A numeric, integer, character or Date vector.
low_serial, high_serial	Bounds, inclusive, on what counts as a plausible serial. The defaults span roughly 1954 to 2078.
year_window	Years that a converted serial may resolve to. A serial landing outside the window is left alone.
date_system	"1900" or "1904"; see excel_date_system().
order	"dm" or "md"; see serial_to_day_month().
output	"numeric" or "character"; see serial_to_day_month(). "character" preserves a trailing zero, so a value typed as 3.10 survives as "3.10" rather than 3.1.
origin_mode	Deprecated. "1900" and "1904" are passed to date_system. "auto" re-enables the 0.1.0 behaviour of picking the date system by comparing candidate dates against ref_date, with a warning: that comparison is a heuristic and can shift every restored value by four years and a day.
ref_date	Reference date used only by origin_mode = "auto".

Value

A vector the same length as x: numeric when every value can be represented as a number, character otherwise.

What is inferred, and what is not

Given a serial, a date system and a field order the conversion is exact (see serial_to_day_month()). Without the file, one thing cannot be known: which numbers *are* serials. That is what low_serial, high_serial and year_window decide, and they are deliberately conservative – a value is converted only if it is a whole number, falls inside the serial range, and resolves to a year inside the window. Everything else is returned untouched.

The other two facts should be supplied rather than inferred. date_system defaults to "1900", which is what every current Excel writes; read the true value with excel_date_system() if you still have the workbook. order defaults to "dm", the locale convention under which 30.3 is turned into a date in the first place.

See Also

unexcel_xlsx() for the file-driven path, fix_serial_columns() for whole data frames.

Examples

```
# 45746 is 2025-03-30; 12.5 is not a whole number, so it is left alone
restore_day_month(c(45746, 12.5, 45823))

# Month-first workbooks
restore_day_month(45746, order = "md")
```

```
# Take the date system from the file rather than the default
path <- system.file("extdata", "typed-numbers.xlsx", package = "unexcel")
restore_day_month(45746, date_system = excel_date_system(path))
```

`serial_to_day_month` *Convert Excel date serials to* `day.month` *values*

Description

The deterministic core of the package: given a serial, a date system and a field order, there is exactly one answer and no inference takes place. Every other function in `unexcel` reduces to this one once it has established those three things – from the workbook itself (`excel_date_system()`, `excel_date_cells()`) when the `.xlsx` file is available, or from the arguments you supply when it is not.

Usage

```
serial_to_day_month(
  x,
  date_system = c("1900", "1904"),
  order = c("dm", "md"),
  output = c("numeric", "character")
)
```

Arguments

`x`	A numeric vector of Excel date serials (or anything coercible to one). Non-finite values and values that are not whole numbers are returned as `NA`, since a fractional serial is a time of day, not a mistyped `day.month`.
`date_system`	Either `"1900"` or `"1904"`. Read it from the workbook with `excel_date_system()` rather than guessing; the default is `"1900"`, which is what every current version of Excel writes.
`order`	`"dm"` for `day.month` (a value typed as `30.3` in a day-first locale) or `"md"` for `month.day`. `unexcel_xlsx()` takes this from each cell's number format code, so it need not be assumed.
`output`	`"numeric"` (the default) returns `30.3`; `"character"` returns `"30.3"`. Use `"character"` when trailing zeros matter – a value typed as `3.10` (3 October) is the number `3.1`, and only the character form can tell it apart from a value typed as `3.1`.

Value

A numeric or character vector the same length as `x`.

See Also

`excel_date_system()` to read the date system from a workbook, `unexcel_xlsx()` to do the whole job from the file.

Examples

```
# 45746 is 2025-03-30 in the 1900 system
serial_to_day_month(45746)

# The same serial means a different date under the 1904 system
serial_to_day_month(45746, date_system = "1904")

# Month-first workbooks
serial_to_day_month(45746, order = "md")

# Keep the trailing zero of a value typed as "3.10"
serial_to_day_month(45933, output = "character")
```

---

unexcel_xlsx *Read an* .xlsx *sheet and undo its date auto-conversion*

Description

Reads a worksheet and restores the values a spreadsheet turned into dates, using only what the file states about itself: the date system from `xl/workbook.xml`, and each cell's number format from `xl/styles.xml`. A cell is restored when the workbook says it is formatted as a date – not when its magnitude happens to fall in a plausible range – so no threshold, year window or reference date is involved, and an ordinary number that happens to look like a serial is never touched.

Usage

```
unexcel_xlsx(
  path,
  sheet = 1,
  col_names = TRUE,
  restore = TRUE,
  order = c("format", "dm", "md"),
  output = c("numeric", "character"),
  na = ""
)
```

Arguments

`path`	Path to an `.xlsx` file.
`sheet`	Sheet name, or its index in the workbook. Defaults to the first sheet.
`col_names`	If `TRUE` (default), the first row of the sheet holds column names, as it would for `utils::read.csv()` or `readxl::read_excel()`.
`restore`	If `TRUE` (default), date-formatted cells come back as `day.month` numerics. If `FALSE`, they come back as `Date` values, giving a plain read of the sheet.
`order`	`"format"` (default) takes the day/month order from each column's number format code, so a `d/m/yyyy` column restores as `day.month` and an `m/d/yyyy` column as `month.day`. Use `"dm"` or `"md"` to override.

output	"numeric" or "character" for the restored columns; see serial_to_day_month().
na	Strings to read as NA.

Details

Compared with restore_day_month(), which works on values already imported and must therefore infer all three of those things, this function has nothing left to infer. Prefer it whenever the original .xlsx is at hand.

Value

A data frame. The unexcel attribute holds the excel_date_columns() report for the columns that were restored, and date_system records the system the workbook declares.

See Also

excel_date_cells() to inspect what would change before changing it; restore_day_month() when only the imported values survive.

Examples

```
path <- system.file("extdata", "typed-numbers.xlsx", package = "unexcel")

# What the spreadsheet stored
unexcel_xlsx(path, restore = FALSE)

# What was typed
unexcel_xlsx(path)

# Which columns were touched, and why
attr(unexcel_xlsx(path), "unexcel")
```

Index

excel_date_cells, 2
excel_date_cells(), *3, 4*, *9*, *11*
excel_date_columns, 3
excel_date_columns(), *2*, *6, 7*, *11*
excel_date_system, 4
excel_date_system(), *6*, *8, 9*
excel_origin, 4
excel_origin(), *4*
excel_sheets, 5

fix_serial_columns, 6
fix_serial_columns(), *8*

restore_day_month, 7
restore_day_month(), *3, 4*, *6*, *11*

serial_to_day_month, 9
serial_to_day_month(), *4*, *6*, *8*, *11*

unexcel_xlsx, 10
unexcel_xlsx(), *2*, *4*, *6–9*
utils::read.csv(), *3*, *10*