Package: tidyods (via r-universe)

December 31, 2024

Title Read cells from ODS files

Version 0.0.1

Description Import cells from ODS files. Identify a cell's postion, value types and formulas, and provide methods to ``rectify'' cells back to a 2-dimensional data.frame.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends R (>= 4.1.0)

Imports cellranger, cli, dplyr (>= 1.1.0), prettyunits, ps, readr, rlang, tibble, tidyr, tidyselect, xml2, zip

URL `https://github.com/mattkerlogue/tidyods`, `https://mattkerlogue.github.io/tidyods/`

BugReports `https://github.com/mattkerlogue/tidyods/issues`

Suggests bench, knitr, lubridate, mockery, palmerpenguins, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev libxml2-dev libx11-dev

Repository https://mattkerlogue.r-universe.dev

RemoteUrl https://github.com/mattkerlogue/tidyods

RemoteRef HEAD

RemoteSha 6a5632456ee3de0c192ece4a145b748f058b6a10

Contents

`ods_sheets` *List the sheets in an ODS file*

Description

Get a list of sheets in the ODS file, either to explore the structure of a file or to use as an input for iterating over a whole spreadsheet document.

Usage

```
ods_sheets(path)
```

Arguments

`path`	The ODS file

Value

A character vector of sheet names

Examples

```
example <- system.file("extdata", "basic_example.ods", package = "tidyods")
ods_sheets(example)
```

`read_ods_cells` *Read cells from an ODS sheet*

Description

Import cells from an OpenDocument Spreadsheet (ODS) file. In the resulting dataset each cell is its own row with columns providing information about the cell's position, value types and containing formulas.

Usage

```
read_ods_cells(path, sheet = 1, quick = FALSE, quiet = FALSE)
```

Arguments

`path`	The ODS file
`sheet`	The sheet name or index number, set to NA for all sheets
`quick`	Whether to use the quick reading process
`quiet`	Whether to silence console messages (recommended for bulk processing)

Details

The aim of `read_ods_cells()` is to extract the constituent value(s) and other information about cells stored in an ODS file and to present that in a "tidy" format that allows further programmatic manipulation. It is modelled after the functionality of `tidyxl::tidy_xlsx()` which performs a similar role for Microsoft Excel spreadsheets.

There are between two to four presentations of a cell's value in the resulting tibble:

- all cells have a `base_value`, a character vector providing the "most raw" version of a cell's value;
- all cells also have a `cell_content`, a character vector providing the version of a cell's value as seen by the user of a spreadsheet application (i.e. having applied number formatting rules);
- for float, currency and percentage value types, cells will have a `numeric_value` with the raw value of the cell as a base R numeric vector, currency value types also have a `currency_symbol` providing the 3-character ISO currency symbol;
- for date value types, cells will have a `date_value`, a character vector with the date or date-time in ISO 8601 format;
- for time value types, cells will have a `time_value`, a character vector with the time duration in ISO 8601 format.

Processing the ODS XML is a memory intensive process, you can achieve significant speed enhancements by setting the `quick` argument to FALSE. This process will extract only a minimum of information about the cells, namely: `sheet`, `row`, `col`, `value_type` and a `base_value`. The `base_value` when using the `quick` argument will combine the raw value stored for float and percentage value types with the `cell_content` (i.e. the formatted character string) for all other value types.

More details on the types of information extracted by `read_ods_cells()`, including examples of how the different value types are stored in the ODS file format and extracted by `read_ods_cells()` can be found in the vignette, `vignette("read_cells", package = "tidyods")`.

Value

A tibble (data.frame) of cells from the ODS sheet(s).

Examples

```
example <- system.file("extdata", "basic_example.ods", package = "tidyods")
example_cells <- read_ods_cells(example, 1)
dplyr::glimpse(example_cells)
```

`read_ods_sheet` *Read an ODS sheet to a rectangular dataset*

Description

A wrapper around `read_ods_cells()` and one of the rectify functions provided in the package.

Usage

```
read_ods_sheet(
  path,
  sheet = 1,
  rectify = c("simple", "smart"),
  skip = 0,
  col_headers = TRUE,
  base_values = TRUE,
  quick = FALSE,
  quiet = FALSE
)
```

Arguments

`path`	The ODS file
`sheet`	The sheet within the ODS file
`rectify`	The method to convert cells to two-dimensions, either "simple" or "smart (see details).
`skip`	The number of rows to skip before attempting to rectify the cells.
`col_headers`	Whether to use the first row (after any skipping) as the column header (`TRUE` is the default), alternatively a character vector of column names can be provided.
`base_values`	Whether to use the base_value of a cell (TRUE, the default) or whether to provide the cell content as seen by a spreadsheet user.
`quick`	Whether to use the quick reading process.
`quiet`	Whether to silence console messages (recommended for bulk processing)

Details

When `rectify = "simple"` then the `simple_rectify()` function will be used to coerce the cells to a sheet. You can instruct the rectifier to skip rows, whether to use the first row as column headers (or provide your own), and whether to use the underlying values or the formatted cell content for the value of the output cell. If `quick = TRUE` this implies using the cell content, thus the user setting for `base_values` is ignored and treated as if set to `FALSE`.

When `rectify = "smart"` then the `smart_rectify()` function will be used, this will attempt to guess the location of the column headers as well as coercing the columns using value type information extracted from the ODS. Using the smart rectifier ignores any settings for `base_values`, `skip` and `col_headers`. You cannot set `quick = TRUE` if you want to use the smart rectifier.

Value

A tibble, presenting cells in a traditional two-dimension spreadsheet format.

Examples

```
example <- system.file("extdata", "basic_example.ods", package = "tidyods")
read_ods_sheet(example, 1)
read_ods_sheet(example, 2, "smart")
```

`simple_rectify` *Convert ODS cells to a two-dimensional dataset*

Description

There are two functions to "rectify" a set of ODS cells extracted by `read_ods_cells()` back into a traditional two-dimensional spreadsheet rectangle.

Usage

```
simple_rectify(
  ods_cells,
  skip = 0,
  col_headers = TRUE,
  values_from = "base_value"
)

smart_rectify(ods_cells)
```

Arguments

`ods_cells`	A set of cells from `read_ods_cells()`.
`skip`	The number of rows to skip before extracting the sheet.
`col_headers`	Whether to use the first row (after any skipping) as the column header (`TRUE` is the default), alternatively a character vector of column names can be provided.
`values_from`	The column from `ods_cells` to output values from.

Details

`simple_rectify()` will perform a basic reshaping of the dataset, by default it will use the `base_value` extracted by `read_ods_cells()` but this can be changed to a different column by setting the `values_from` argument. The rectifier will also by default try to use the first row for column headers, alternatively you can provide your own column names or set `col_headers = FALSE` to get generic column names of `x1`, `x2`, etc.

`smart_rectify()` performs a more complex reshaping of the dataset, by guessing the location of column headers and using the `value_type` information generated by `read_ods_cells()` to determine whether columns can be coerced to a non-string data type (either numeric, logical, date or time).

Value

A tibble representing the original spreadsheet format.

Examples

```
example <- system.file("extdata", "basic_example.ods", package = "tidyods")
example_cells <- read_ods_cells(example, 1)
simple_rectify(example_cells)
smart_rectify(example_cells)
```

Index