Package 'SheetReader'

October 12, 2022

Type Package

Title Parse xlsx Files

Version 1.0.2

Date 2022-04-07

Description Uses C++ via the 'Rcpp' package to parse modern Excel files ('.xlsx').
Memory usage is kept minimal by decompressing only parts of the file at a time,
while employing multiple threads to achieve significant runtime reduction.
Uses <https://github.com/richgel999/miniz>, <https://github.com/ebiggers/libdeflate>,
and <https://github.com/lemire/fast_double_parser>.

License MIT + file LICENSE

Imports Rcpp (>= 1.0.5)

LinkingTo Rcpp

URL https://github.com/fhenz/SheetReader-r

BugReports https://github.com/fhenz/SheetReader-r/issues

Encoding UTF-8

NeedsCompilation yes

Author Felix Henze [aut, cre],
Rich Geldreich [ctb, cph] (Author of included miniz code),
Eric Biggers [ctb, cph] (Author of included libdeflate code),
Daniel Lemire [ctb, cph] (Author of included fast_double_parser code)

Maintainer Felix Henze <felixhenze0@gmail.com>

Repository CRAN

Date/Publication 2022-04-08 14:12:39 UTC

R topics documented:

SheetReader-package *Fast and efficient xlsx parsing*

Description

Uses C++ via the 'Rcpp' package to parse modern Excel files ('.xlsx'). Memory usage is kept minimal by decompressing only parts of the file at a time, while employing multiple threads to achieve significant runtime reduction.

Details

The only function provided by this package is `read_xlsx()`, with options to determine parsing behaviour.

Author(s)

Felix Henze

Maintainer: Felix Henze <felixhenze0@gmail.com>

read_xlsx *Parse data from a xlsx file*

Description

Parse tabular data from a sheet inside a xlsx file into a data.frame

Usage

```
read_xlsx(
    path,
    sheet = NULL,
    headers = TRUE,
    skip_rows = 0,
    skip_columns = 0,
    method = "efficient",
    num_threads = -1
)
```

Arguments

path	The path to the xlsx file that is to be parsed.
sheet	Which sheet in the file to parse. Can be either the index/position (1 = first sheet) or name. By default parses the first sheet.
headers	Whether to interpret the first row as column names.

skip_rows	How many rows should be skipped before values are read.
skip_columns	How many columns should be skipped before values are read.
method	What method should be used to parse the file. Can be either "efficient" or "fast". See 'Details' for more information.
num_threads	The number of threads to use for parsing. Will be automatically determined if not provided. Note that the "efficient" method uses a minimum of 2 threads even if only 1 is specified.

Details

2 methods for parsing are provided; "efficient" and "fast".

The "efficient" method achieves minimal memory usage by only ever decompressing a small part of the file. This part is parsed and then replaced with the next decompressed content. Decompression and parsing are done by separate threads (parsing is additionally divided among multiple threads) to achieve good runtimes. Unfortunately, currently using multiple parse threads relies on cells also storing their position in the sheet within themselves. This is done by Excel itself, but for other xlsx-generating tools this may not be guaranteed. In these cases only 1 parse thread can be used.

The "fast" method decompresses the whole file into memory using the full-buffer optimized 'libdeflate' library. This is faster than the "efficient" method but obviously requires more memory. If you are sure you have enough memory you can specify this method to get slightly better runtime (be aware that the decompressed content can be significantly larger than the shown file size).

Value

data.frame

Examples

```
exampleFile <- system.file("extdata", "multi-test.xlsx", package = "SheetReader")

# Read first sheet of the file, using first row as column names
df1 <- read_xlsx(exampleFile, sheet = 1, headers = TRUE)
head(df1)

# Read the "encoding" sheet, skipping 1 row and not using the next row as column names
df2 <- read_xlsx(exampleFile, sheet = "encoding", headers = FALSE, skip_rows = 1)
head(df2)
```

Index