Package 'hipread'

November 30, 2023

Type Package

Title Read Hierarchical Fixed Width Files

Version 0.2.4

Contact ipums@umn.edu

Description Read hierarchical fixed width files like those commonly used by many census data providers. Also allows for reading of data in chunks, and reading 'gzipped' files without storing the full file in memory.

License GPL (>= 2) | file LICENSE

Encoding UTF-8

LinkingTo Rcpp, BH

Imports Rcpp, R6, rlang, tibble

Suggests dplyr, readr, testthat

RoxygenNote 7.2.3

NeedsCompilation yes

Author Greg Freedman Ellis [aut],
Derek Burk [aut, cre],
Joe Grover [ctb],
Mark Padgham [ctb],
Hadley Wickham [ctb] (Code adapted from readr),
Jim Hester [ctb] (Code adapted from readr),
Romain Francois [ctb] (Code adapted from readr),
R Core Team [ctb] (Code adapted from readr),
RStudio [cph, fnd] (Code adapted from readr),
Jukka Jylänki [ctb, cph] (Code adapted from readr),
Mikkel Jørgensen [ctb, cph] (Code adapted from readr),
University of Minnesota [cph]

Maintainer Derek Burk <ipums+cran@umn.edu>

Repository CRAN

Date/Publication 2023-11-30 15:50:02 UTC

R topics documented:

`hipread_example` *Get path to hipread's example datasets*

Description

Get access to example extracts.

Usage

```
hipread_example(path = NULL)
```

Arguments

`path`	Name of file. If `NULL`, the example files will be listed.

Value

The filepath to an example file, or if path is empty, a vector of all available files.

Examples

```
hipread_example() # Lists all available examples
hipread_example("test-basic.dat") # Gives filepath for a basic example
```

`hipread_long` *Read a hierarchical fixed width data file*

Description

Analogous to `readr::read_fwf()` but allowing for hierarchical fixed width data files (where the data file has rows of different record types, each with their own variables and column specifications). `hipread_long()` reads hierarchical data into "long" format, meaning that there is one row per observation, and variables that don't apply to the current observation receive missing values. Alternatively, `hipread_list()` reads hierarchical data into "list" format, which returns a list that has one data.frame per record type.

Usage

```
hipread_long(
  file,
  var_info,
  rt_info = hip_rt(1, 0),
  compression = NULL,
  skip = 0,
  n_max = -1,
  encoding = "UTF-8",
  progress = show_progress()
)

hipread_list(
  file,
  var_info,
  rt_info = hip_rt(1, 0),
  compression = NULL,
  skip = 0,
  n_max = -1,
  encoding = "UTF-8",
  progress = show_progress()
)
```

Arguments

file	A filename
var_info	Variable information, specified by either hip_fwf_positions() or hip_fwf_widths(). For hierarchical data files, there should be a named list, where the name is the value indicated by the record type variable and there is one variable information per record type.
rt_info	A record type information object, created by hip_rt(), which contains information about the location of the record type variable that defines the record type for each observation. The default contains width 0, which indicates that there the data is rectangular and does not have a record type variable.
compression	If NULL, guesses the compression from the file extension (if extension is "gz" uses gzip, otherwise treats as plain text), can specify it with a string ("txt" indicates plain text and "gz" for gzip).
skip	Number of lines to skip at the start of the data (defaults to 0).
n_max	Maximum number of lines to read. Negative numbers (the default) reads all lines.
encoding	(Defaults to UTF-8) A string indicating what encoding to use when reading the data, but like readr, the data will always be converted to UTF-8 once it is imported. Note that UTF-16 and UTF-32 are not supported for non-character columns.
progress	A logical indicating whether progress should be displayed on the screen, defaults to showing progress unless the current context is non-interactive or in a knitr

document or if the user has turned off readr's progress by default using the option `options("readr.show_progress")`.

Value

A `tbl_df` data frame

Examples

```
# Read an example hierarchical data.frame into long format
data <- hipread_long(
  hipread_example("test-basic.dat"),
  list(
    H = hip_fwf_positions(
      c(1, 2, 5, 8),
      c(1, 4, 7, 10),
      c("rt", "hhnum", "hh_char", "hh_dbl"),
      c("c", "i", "c", "d")
    ),
    P = hip_fwf_widths(
      c(1, 3, 1, 3, 1),
      c("rt", "hhnum",  "pernum", "per_dbl", "per_mix"),
      c("c", "i", "i", "d", "c")
    )
  ),
  hip_rt(1, 1)
)

# Read an example hierarchical data.frame into list format
data <- hipread_list(
  hipread_example("test-basic.dat"),
  list(
    H = hip_fwf_positions(
      c(1, 2, 5, 8),
      c(1, 4, 7, 10),
      c("rt", "hhnum", "hh_char", "hh_dbl"),
      c("c", "i", "c", "d")
    ),
    P = hip_fwf_widths(
      c(1, 3, 1, 3, 1),
      c("rt", "hhnum",  "pernum", "per_dbl", "per_mix"),
      c("c", "i", "i", "d", "c")
    )
  ),
  hip_rt(1, 1)
)

# Read a rectangular data.frame
data_rect <- hipread_long(
  hipread_example("test-basic.dat"),
  hip_fwf_positions(
    c(1, 2),
    c(1, 4),
```

```
    c("rt", "hhnum"),
    c("c", "i")
  )
)
```

`hipread_long_chunked` *Read a hierarchical fixed width data file, in chunks*

Description

Analogous to `readr::read_fwf()`, but with chunks, and allowing for hierarchical fixed width data files (where the data file has rows of different record types, each with their own variables and column specifications). `hipread_long_chunked()` reads hierarchical data into "long" format, meaning that there is one row per observation, and variables that don't apply to the current observation receive missing values. Alternatively, `hipread_list_chunked()` reads hierarchical data into "list" format, which returns a list that has one data.frame per record type.

Usage

```
hipread_long_chunked(
  file,
  callback,
  chunk_size,
  var_info,
  rt_info = hip_rt(1, 0),
  compression = NULL,
  skip = 0,
  encoding = "UTF-8",
  progress = show_progress()
)

hipread_list_chunked(
  file,
  callback,
  chunk_size,
  var_info,
  rt_info = hip_rt(1, 0),
  compression = NULL,
  skip = 0,
  encoding = "UTF-8",
  progress = show_progress()
)
```

Arguments

`file`	A filename
`callback`	A `callback` function, allowing you to perform a function on each chunk.

chunk_size	The size of the chunks that will be read as a single unit (defaults to 10000)
var_info	Variable information, specified by either hip_fwf_positions() or hip_fwf_widths(). For hierarchical data files, there should be a named list, where the name is the value indicated by the record type variable and there is one variable information per record type.
rt_info	A record type information object, created by hip_rt(), which contains information about the location of the record type variable that defines the record type for each observation. The default contains width 0, which indicates that there the data is rectangular and does not have a record type variable.
compression	If NULL, guesses the compression from the file extension (if extension is "gz" uses gzip, otherwise treats as plain text), can specify it with a string ("txt" indicates plain text and "gz" for gzip).
skip	Number of lines to skip at the start of the data (defaults to 0).
encoding	(Defaults to UTF-8) A string indicating what encoding to use when reading the data, but like readr, the data will always be converted to UTF-8 once it is imported. Note that UTF-16 and UTF-32 are not supported for non-character columns.
progress	A logical indicating whether progress should be displayed on the screen, defaults to showing progress unless the current context is non-interactive or in a knitr document or if the user has turned off readr's progress by default using the option options("readr.show_progress").

Value

Depends on the type of callback function you use

Examples

```
# Read in a data, filtering out hhnum == "002"
data <- hipread_long_chunked(
  hipread_example("test-basic.dat"),
  HipDataFrameCallback$new(function(x, pos) x[x$hhnum != 2, ]),
  4,
  list(
    H = hip_fwf_positions(
      c(1, 2, 5, 8),
      c(1, 4, 7, 10),
      c("rt", "hhnum", "hh_char", "hh_dbl"),
      c("c", "i", "c", "d")
    ),
    P = hip_fwf_widths(
      c(1, 3, 1, 3, 1),
      c("rt", "hhnum",  "pernum", "per_dbl", "per_mix"),
      c("c", "i", "i", "d", "c")
    )
  ),
  hip_rt(1, 1)
)
```

`hipread_long_yield` *Read a hierarchical fixed width data file, in yields*

Description

Enhances `hipread_long()` or `hipread_list()` to allow you to read hierarchical data in pieces (called 'yields') and allow your code to have full control between reading pieces, allowing for more freedom than the 'callback' method introduced in the chunk functions (like `hipread_long_chunked()`).

Usage

```
hipread_long_yield(
  file,
  var_info,
  rt_info = hip_rt(1, 0),
  compression = NULL,
  skip = 0,
  encoding = "UTF-8"
)

hipread_list_yield(
  file,
  var_info,
  rt_info = hip_rt(1, 0),
  compression = NULL,
  skip = 0,
  encoding = "UTF-8"
)
```

Arguments

`file`	A filename
`var_info`	Variable information, specified by either `hip_fwf_positions()` or `hip_fwf_widths()`. For hierarchical data files, there should be a named list, where the name is the value indicated by the record type variable and there is one variable information per record type.
`rt_info`	A record type information object, created by `hip_rt()`, which contains information about the location of the record type variable that defines the record type for each observation. The default contains width 0, which indicates that there the data is rectangular and does not have a record type variable.
`compression`	If `NULL`, guesses the compression from the file extension (if extension is "gz" uses gzip, otherwise treats as plain text), can specify it with a string ("txt" indicates plain text and "gz" for gzip).
`skip`	Number of lines to skip at the start of the data (defaults to 0).

encoding (Defaults to UTF-8) A string indicating what encoding to use when reading the data, but like readr, the data will always be converted to UTF-8 once it is imported. Note that UTF-16 and UTF-32 are not supported for non-character columns.

Details

These functions return a HipYield R6 object which have the following methods:

- `yield(n = 10000)` A function to read the next 'yield' from the data, returns a `tbl_df` (or list of `tbl_df` for `hipread_list_yield()`) with up to n rows (it will return NULL if no rows are left, or all available ones if less than n are available).
- `reset()` A function to reset the data so that the next yield will read data from the start.
- `is_done()` A function that returns whether the file has been completely read yet or not.
- `cur_pos` A property that contains the next row number that will be read (1-indexed).

Value

A HipYield R6 object (See 'Details' for more information)

Methods

Public methods:

- `HipYield$new()`
- `HipYield$yield()`
- `HipYield$reset()`
- `HipYield$is_done()`

Method `new()`:

Usage:

```
HipYield$new(
  file,
  var_info,
  rt_info = hip_rt(0, 1),
  compression = NULL,
  skip = 0,
  encoding = NULL
)
```

Method `yield()`:

Usage:

```
HipYield$yield(n = 10000)
```

Method `reset()`:

Usage:

```
HipYield$reset()
```

Method `is_done()`:

Usage:

```
HipYield$is_done()
```

Super class

`hipread::HipYield` -> `HipLongYield`

Methods

Public methods:

- `HipLongYield$new()`
- `HipLongYield$yield()`

Method `new()`:

Usage:

```
HipLongYield$new(
  file,
  var_info,
  rt_info = hip_rt(0, 1),
  compression = NULL,
  skip = 0,
  encoding = NULL
)
```

Method `yield()`:

Usage:

```
HipLongYield$yield(n = 10000)
```

Super class

`hipread::HipYield` -> `HipListYield`

Methods

Public methods:

- `HipListYield$new()`
- `HipListYield$yield()`

Method `new()`:

Usage:

```
HipListYield$new(
  file,
  var_info,
  rt_info = hip_rt(0, 1),
  compression = NULL,
  skip = 0,
  encoding = NULL
)
```

Method `yield()`:

Usage:

```
HipListYield$yield(n = 10000)
```

Examples

```
library(hipread)
data <- hipread_long_yield(
  hipread_example("test-basic.dat"),
  list(
    H = hip_fwf_positions(
      c(1, 2, 5, 8),
      c(1, 4, 7, 10),
      c("rt", "hhnum", "hh_char", "hh_dbl"),
      c("c", "i", "c", "d")
    ),
    P = hip_fwf_widths(
      c(1, 3, 1, 3, 1),
      c("rt", "hhnum",  "pernum", "per_dbl", "per_mix"),
      c("c", "i", "i", "d", "c")
    )
  ),
  hip_rt(1, 1)
)
# Read the first 4 rows
data$yield(4)

# Read the next 2 rows
data$yield(2)

# Reset and then read the first 4 rows again
data$reset()
data$yield(4)
```

`hip_fwf_positions` *Specify column-specific options for hipread*

Description

Specify column specifications analogous to `readr::fwf_positions()`. However, unlike in readr, the column type information is specified alongside the column positions and there are two extra options that can be specified (`trim_ws` gives control over trimming whitespace in character columns, and `imp_dec` allows for implicit decimals in double columns).

Usage

```
hip_fwf_positions(
  start,
  end,
  col_names,
  col_types,
  trim_ws = TRUE,
  imp_dec = 0
)
```

```
hip_fwf_widths(widths, col_names, col_types, trim_ws = TRUE, imp_dec = 0)
```

Arguments

start, end	A vector integers describing the start and end positions of each field
col_names	A character vector of variable names
col_types	A vector of column types (specified as either "c" or "character" for character, "d" or "double" for double and "i" or "integer" for integer).
trim_ws	A logical vector, indicating whether to trim whitespace on both sides of character columns (Defaults to TRUE, ignored on non-character columns).
imp_dec	An integer vector, indicating the number of implicit decimals on a double variable (Defaults to 0, ignored on non-double columns).
widths	A vector of integer widths for each field (assumes that columns are consecutive - that there is no overlap or gap between fields)

Value

A data.frame containing the column specifications

Examples

```
# 3 Columns, specified by position
hip_fwf_positions(
  c(1, 3, 7),
  c(2, 6, 10),
  c("Var1", "Var2", "Var3"),
  c("c", "i", "d")
)

# The same 3 columns, specified by width
hip_fwf_widths(
  c(2, 4, 4),
  c("Var1", "Var2", "Var3"),
  c("c", "i", "d")
)
```

---

hip_rt *Create a record type information object*

Description

Create a record type information object for hipread to use when reading hierarchical files. A width of 0 indicates that the file is rectangular (eg a standard fixed width file).

Usage

```
hip_rt(start, width, warn_on_missing = TRUE)
```

Arguments

start	Start position of the record type variable
width	The width of the record type variable
warn_on_missing	Whether to warn when encountering a record type that is not specified

Value

A list, really only intended to be used internally by hipread

Index