Package: chunkR (via r-universe)

November 5, 2024

Type Package

Title Read Tables in Chunks

Version 1.1.1

Date 2018-02-21

Maintainer Leandro Roser <learoser@gmail.com>

Description Read tables chunk by chunk using a C++ backend and a simple R interface.

License GPL (>= 2)

LazyLoad yes

Depends R (>= 3.0), methods

Suggests testthat, covr

Imports Rcpp (>= 0.12.14)

LinkingTo Rcpp

Collate 'ZZZ.R' 'auxiliary.R' 'classes.R' 'constructors.R' 'deprecated.R' 'methods.R' 'RcppExports.R'

RoxygenNote 6.0.1

NeedsCompilation yes

Author Leandro Roser [aut, cre]

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

RemoteUrl https://github.com/leandroroser/chunkr

RemoteRef HEAD

RemoteSha 3f9ab1a48f596622cc392116e4a42f0f43182658

Contents

---

chunkR-package *Read Tables in Chunks*

Description

Read tables chunk by chunk using a C++ backend and a simple R interface.

Details

This package allows to read long text tables in chunks. The objects of class "chunker" are the central elements of the chunkR package. These objects can store a data chunk and other information required for the process of reading a file in pieces. A chunker object is created with the `chunker` function, that requires the path to a file, and other arguments, as the size of the chunk. Two basic methods are defined to manipulate the object:

- `next_chunk` function to read the next chunk

- `get_table` to retrieve the data

The functions `get_completed` and `get_colnames` allow to get the number of rows already read, and the column names of the table.

The program can read data frames (with column type detection) or matrices. The program reads each chunk with the `next_chunk` function (that return TRUE), and makes it accessible via the `get_table` function. After reading all the file, `next_chunk` function returns FALSE and the `get_table` function an empty data frame/matrix.

Author(s)

Leandro Roser

Maintainer: Leandro Roser <learoser@gmail.com>

Examples

```
data(iris)

# write iris as tab delimited file. Note that quote is set to FALSE

tmp_path <- file.path(tempdir(),"iris.txt")
write.table(iris, tmp_path, quote = FALSE)

#-----------------------------------------------------------------#
#--- Reading a data frame with automatic column-type detection ---#
#-----------------------------------------------------------------#

# create a 'chunker' object passing the path of the input file.
my_chunker_object <- chunker(tmp_path, chunksize = 30)

# read a chunk
next_chunk(my_chunker_object)
```

```
# get the chunk
 get_table(my_chunker_object)

# read another chunk
next_chunk(my_chunker_object)

# get the number of lines already read
get_completed(my_chunker_object)


#--- read a csv file ---#

tmp_path_csv <- file.path(tempdir(),"iris.csv")

write.table(iris, tmp_path_csv, quote = FALSE, sep = ",")

# read the csv indicating the value of the sep parameter
my_chunker_object2 <- chunker(tmp_path_csv, chunksize = 30, sep = ",")
# the file can  then be processed as with tab delimiters

# remove temporal file
file.remove(tmp_path_csv)

#-----------------------------------------------------#
#--- Reading a data frame using column types argument ---#
#-----------------------------------------------------#

## Four types can be passed : "character", "numeric" (aka "double"), "integer", "logical"

# create a 'chunker' object passing the path of the input file.
my_chunker_object3 <- chunker(tmp_path, chunksize = 120,
 columns_classes = c("numeric", "numeric", "numeric","numeric", "character"))

# read a chunk
next_chunk(my_chunker_object3)

# get the chunk
get_table(my_chunker_object3)

# read another chunk
next_chunk(my_chunker_object3)

# get the number of lines already read
get_completed(my_chunker_object3)


#-------------------------#
#--- Reading a matrix  ---#
#-------------------------#

my_chunker_object4 <- chunker(tmp_path, chunksize = 30, data_format= "matrix")
```

```
# store the chunk as a character matrix in R
this_data <- get_table(my_chunker_object4)


# The package provides a fast generic C++ function for conversion from
# matrix (any R type) to data frame
this_data_as_df2 <- matrix2df(this_data)

# remove temporal file
file.remove(tmp_path)


 ## Not run:
#----------------------------------#
#--- Example with a big table -----#
#----------------------------------#

### Example with a data frame

# create a large data frame, and write it in a temporal directory

tmp_path <- file.path(tempdir(),"big_table.txt")

out <- data.frame(numeric_data = runif(1000000),
                  character_data = sample(c("a", "t", "c", "g"), 1000000,
                  replace = TRUE),
                  integer_data = sample(1000000),
                  bool_data = sample(c(TRUE, FALSE), 1000000, replace = TRUE))


write.table(out, tmp_path, quote = FALSE)

# create a chunker object, reading in chunks of 10000 lines
my_chunker_object5 <- chunker(tmp_path, chunksize = 10000)

next_chunk(my_chunker_object5)
data <- get_table(my_chunker_object5)

# check classes
lapply(data,typeof)
file.remove(tmp_path)


### Example with a matrix

# create a large matrix, and write it in a temporal directory

my_table <- tempfile()
write.table(matrix(sample(c("a", "t", "c", "g"), 1000000, replace = TRUE),
100000, 1000), my_table, quote = FALSE)

# create a chunker object, reading in chunks of 10000 lines
my_chunker_object6 <- chunker(my_table, chunksize = 10000, data_format= "matrix")
```

```
# create a loop to read all the file and make something with it

lines <- 0
while(next_chunk(my_chunker_object6))
{
  data <- get_table(my_chunker_object6)

  # do something with data, e.g., convert to data frame first
  data <- matrix2df(data)

  lines <- lines + nrow(data)
  cat("Processed ", lines, "lines\n")
}

# remove the temporal file
file.remove(my_table)

## End(Not run)
```

---

chunker *chunker*

Description

The objects of class "chunker" are the central elements of the chunkR package. These objects can store a data chunk and other information required for the process of reading a file in pieces. A "chunker" object is created with the chunker() function, that requires the path to a file, and other arguments, as the size of the chunk and the data type ("data.frame" or "matrix"). Two basic methods are defined to manipulate the object:

- next_chunk function to read the next chunk

- get_table function to retrieve the data

The functions get_completed and get_colnames allow to get the number of rows already read, and the column names of the table.

Usage

```
chunker(path, sep = " ", quoted = FALSE, has_colnames = TRUE,
  has_rownames = TRUE, chunksize = 1000L, data_format = c("data.frame",
  "matrix"), columns_classes = character(0), autodetect = TRUE,
  scan_rows = 10)
```

Arguments

path	Input file path
sep	Character separating cells in the input table (default = " ")
quoted	Quoted character data? Default FALSE. If TRUE, the program removes quotes.

has_colnames	Column names present in the input table? (Logical, default TRUE)
has_rownames	Row names present in the input table? (Logical, default TRUE)
chunksize	Chunk size (default 1000)
data_format	Format of input data: "data.frame" (default) or "matrix".
columns_classes	Vector with the class of each column: "character", "numeric" (aka "double"), "integer" or "logical".
autodetect	Use auto-detection of columns classes? Default TRUE.
scan_rows	How many rows to scan for auto-detection of columns classes. Default is 10. Note that this value shoud be increased when columns only have NA values in the scanned rows. Columns classes are detected via a call to read.table with the scan_rows value passed to the nrows parameter.

Examples

```
data(iris)

# write iris as tab delimited file. Note that quote is set to FALSE

tmp_path <- file.path(tempdir(),"iris.txt")
write.table(iris, tmp_path, quote = FALSE)

#---------------------------------------------------------------------#
#--- Reading a data frame with automatic column-type detection ---#
#---------------------------------------------------------------------#

# create a 'chunker' object passing the path of the input file.
my_chunker_object <- chunker(tmp_path, chunksize = 30)

# read a chunk
next_chunk(my_chunker_object)

# get the chunk
 get_table(my_chunker_object)

# read another chunk
next_chunk(my_chunker_object)

# get the number of lines already read
get_completed(my_chunker_object)


#--- read a csv file ---#

tmp_path_csv <- file.path(tempdir(),"iris.csv")

write.table(iris, tmp_path_csv, quote = FALSE, sep = ",")

# read the csv indicating the value of the 'sep' parameter
```

```
my_chunker_object2 <- chunker(tmp_path_csv, chunksize = 30, sep = ",")
# the file can  then be processed as with tab delimiters

# remove temporal file
file.remove(tmp_path_csv)

#-----------------------------------------------------------#
#--- Reading a data frame using column types argument ---#
#-----------------------------------------------------------#

## Four types can be passed : "character", "numeric" (aka "double"), "integer", "logical"

# create a 'chunker' object passing the path of the input file.
my_chunker_object3 <- chunker(tmp_path, chunksize = 120,
 columns_classes = c("numeric", "numeric", "numeric","numeric", "character"))

# read a chunk
next_chunk(my_chunker_object3)

# get the chunk
get_table(my_chunker_object3)

# read another chunk
next_chunk(my_chunker_object3)

# get the number of lines already read
get_completed(my_chunker_object3)


#-------------------------#
#--- Reading a matrix  ---#
#-------------------------#

my_chunker_object4 <- chunker(tmp_path, chunksize = 30, data_format= "matrix")

# store the chunk as a character matrix in R
this_data <- get_table(my_chunker_object4)


# The package provides a fast generic C++ function for conversion from
# matrix (any R type) to data frame
this_data_as_df2 <- matrix2df(this_data)

# remove temporal file
file.remove(tmp_path)

## Not run:

#---------------------------------#
#--- Example with a big table -----#
#---------------------------------#

### Example with a data frame
```

```
# create a large data frame, and write it in a temporal directory

tmp_path <- file.path(tempdir(),"big_table.txt")

out <- data.frame(numeric_data = runif(1000000),
                  character_data = sample(c("a", "t", "c", "g"), 1000000,
                  replace = TRUE),
                  integer_data = sample(1000000),
                  bool_data = sample(c(TRUE, FALSE), 1000000, replace = TRUE))


write.table(out, tmp_path, quote = FALSE)

# create a chunker object, reading in chunks of 10000 lines
my_chunker_object5 <- chunker(tmp_path, chunksize = 10000)

next_chunk(my_chunker_object5)
data <- get_table(my_chunker_object5)

# check classes
lapply(data,typeof)
file.remove(tmp_path)


### Example with a matrix

# create a large matrix, and write it in a temporal directory

my_table <- tempfile()
write.table(matrix(sample(c("a", "t", "c", "g"), 1000000, replace = TRUE),
100000, 1000), my_table, quote = FALSE)

# create a chunker object, reading in chunks of 10000 lines
my_chunker_object6 <- chunker(my_table, chunksize = 10000, data_format= "matrix")

# create a loop to read all the file and make something with it

lines <- 0
while(next_chunk(my_chunker_object6))
{
  data <- get_table(my_chunker_object6)

  # do something with data, e.g., convert to data frame first
  data <- matrix2df(data)

  lines <- lines + nrow(data)
  cat("Processed ", lines, "lines\n")
}

# remove the temporal file
file.remove(my_table)
```

```
## End(Not run)
```

`chunker methods` *Manipulation methods for chunker objects*

Description

chunker objects can be manipulated with the following methods:

1. **next_chunk**: allows to read the next chunk of a chunker object
2. **get_table**: retrieve the current data chunk contained in the object

In addition, the following information can be retrieved from chunker objects:

1. **get_completed**: get the number of rows already read
2. **get_colnames**: get column names of the chunker object

Usage

```
next_chunk(obj)

## S4 method for signature 'chunker'
next_chunk(obj)

get_table(obj)

## S4 method for signature 'chunker'
get_table(obj)

get_colnames(obj)

## S4 method for signature 'chunker'
get_colnames(obj)

get_completed(obj)

## S4 method for signature 'chunker'
get_completed(obj)

get_total(obj)

## S4 method for signature 'chunker'
get_total(obj)

get_type(obj)
```

```
## S4 method for signature 'chunker'
get_type(obj)

get_attr(obj)

## S4 method for signature 'chunker'
get_attr(obj)
```

Arguments

obj	object of class chunker

Details

See chunker for examples.

`chunkR_devel` *chunkR devel*

Description

The function opens the chunkR-devel web site: https://github.com/leandroroser/chunkR

Usage

```
chunkR_devel()
```

`matrix2df` *matrix2df*

Description

conversion from matrix to DataFrame

Arguments

x	matrix

Index