Package 'bread'

January 19, 2022

Title Analyze Big Files Without Loading Them in Memory

Version 0.1.7

Description A simple set of wrapper functions for data.table::fread() that allows subsetting or filtering rows and selecting columns of table-formatted files too large for the available RAM. 'b stands for 'big files'. The package is using Unix commands like grep, cut and sed through (hopefully) intuitive parameters.
bread makes heavy use of Unix commands like grep, sed, wc and cut. They are available by default in all Unix environments.
For Windows, you need to install those commands externally in order to simulate a Unix environment and make sure that the executables are in the Windows PATH variable.
To my knowledge, the simplest ways are to install RTools, Git or Cygwin. If they have been correctly installed (with the expected registry entries), they should be detected on loading the package and the correct directories will be added automatically to the PATH.

License GPL (>= 3)

URL https://github.com/MagicHead99/bread/

BugReports https://github.com/MagicHead99/bread/issues

Encoding UTF-8

RoxygenNote 7.1.2

Depends dplyr

Imports data.table, stringr

NeedsCompilation no

Author Vincent Guegan [aut, cre]

Maintainer Vincent Guegan <vincent.guegan@banque-france.fr>

Repository CRAN

Date/Publication 2022-01-19 15:22:56 UTC

R topics documented:

`bcolnames` *Retrieve the column names directly from a big file without loading it in memory*

Description

Simply reads the first line of a file with data.table::fread and the head Unix command. This allows analyzing big files that would not fit in memory (and cause an error such as "cannot allocate vector of size").

Usage

```
bcolnames(file = NULL, ...)
```

Arguments

`file`	String. Name or full path to a file compatible with data.table::fread()
`...`	Arguments that must be passed to data.table::fread() like "sep".

Value

A character vector

Examples

```
file <- system.file("extdata", "test.csv", package = "bread")
## Retrieving the column names
bcolnames(file = file)
```

`bfile_split` *Splits a big file in several smaller files without loading it entirely in memory*

Description

This function helps splitting a big csv file in smaller csv files using one of those 3 methods:

1. by_nrows: Each new file will contain a number of rows defined by the user
2. by_nfiles: The user decide the number of files created with the rows equally distributed
3. by_columns: The file will be split by the combinations of unique values in the columns chosen by the user Like all other functions in the bread package, this is achieved using Unix commands that allow opening, reading and splitting big files that wouldn't fit in memory (The goal being to help with the "cannot allocate vector of size" error).

Usage

```
bfile_split(
  file = NULL,
  by_nfiles,
  by_nrows,
  by_columns,
  drop_empty_files = T,
  write_sep = NA,
  write_dir = NULL,
  meta_output = NULL,
  ...
)
```

Arguments

`file`	String. Name or full path to a file compatible with data.table::fread()
`by_nfiles`	Numeric. Number of files with an equal number of rows to be created. Only the last one will be slightly larger, containing the remainder.
`by_nrows`	Numeric. Number of rows composing the new split files. The last one may be smaller, containing only the remainder.
`by_columns`	Vector of strings or numeric. Indicates either the names or index number of the columns whose combinations of unique values will be used to split the files.
`drop_empty_files`	Logical. Defaults to TRUE. Used only with the "by_column" argument. If changed to FALSE, empty files may be created.
`write_sep`	One character-length string. Will be provided to data.table::fwrite() for writing the output. If not provided, the delimiter will be guessed from the input file with the bsep() function

write_dir	String. Path to the output directory. By default, it will be the working directory. If the directory doesn't exist, it will be created.
meta_output	List. Optional. Output of the bmeta() function on the same file. It indicates the names and numbers of columns and rows. If not provided, it will be calculated. It can take a while on file with several million rows.
...	Arguments that must be passed to data.table::fread() like "sep=" and "dec=".

Value

Creates a number of csv files from the original larger file

Examples

```
## Not run:
file <- system.file("extdata", "test.csv", package = "bread")
## Filtering on 2 columns, using regex.
bfile_split(file = file, by_nrows = 5)
bfile_split(file = file, by_nfiles = 3)
bfile_split(file = file, by_columns = c("YEAR", "COLOR"))
## For very big files with several million rows, the bmeta() function takes
##a long time to count the rows without loading the file in memory.
## Best practice is to save the result of bmeta() in a variable and provide it
## to bfile_split()
meta <- bmeta(file = file)
bfile_split(file = file, by_nrows = 5, meta_output = meta)
## write_sep can be used to write the output files with a different delimiters than the input file
bfile_split(file = file, by_nrows = 5, write_sep = "*")

## End(Not run)
```

---

bfilter *Pre-filters a data file using column values before loading it in memory*

Description

Simple wrapper for data.table::fread() allowing to filter data from a file with the Unix grep command. This method is useful if you want to load a file too large for your available memory (and encounter the "cannot allocate vector of size" error for example).

Usage

```
bfilter(
  file = NULL,
  patterns = NULL,
  filtered_columns = NULL,
  fixed = FALSE,
  ...
)
```

Arguments

file	String. Name or full path to a file compatible with data.table::fread()
patterns	Vector of strings. One or several patterns used to filter the data from the input file. Each element of the vector should correspond to the column to be filtered. Can use regular expressions.
filtered_columns	Vector of strings or numeric. The columns to be filtered should be indicated through their names or their index number. Each element of the vector should correspond to the pattern with which it will be filtered.
fixed	Logical. If TRUE, pattern is a string to be matched as is. Overrides all conflicting arguments.
...	Arguments that must be passed to data.table::fread() like "sep" and "dec".

Value

A dataframe

Examples

```
file <- system.file("extdata", "test.csv", package = "bread")
## Filtering on 2 columns, using regex.
bfilter(file = file, patterns = c("200[4-6]", "red"),
      filtered_columns = c("YEAR", "COLOR"), sep = ";")
bfilter(file = file, patterns = c("2004|2005", "red"),
      filtered_columns = c("YEAR", "COLOR"), sep = ";")
## You need to use fixed = T if some patterns contain special characters
## that mess with regex like "(" and ")"
bfilter(file = file, patterns = "orange (purple)",
      filtered_columns = "COLOR", fixed = TRUE, sep = ";")
## If you do not provide the filtered_columns, you risk encountering
## false positives because the grep command filters on the whole file,
## not column by column. Here, the value 2002 will be found in the "PRICE"
## column as well. The filtered_column argument will just make the script
## do a second pass with dplyr::filter() to remove false positives.
bfilter(file = file, patterns = "2002", sep = ";")
```

---

bmeta *Helper function generating nrow and colnames for the target file without loading it in memory*

Description

Generates a list that can be passed to the bfile_split() function in order to indicate the number of rows and the name of columns in the file. The output can be saved in a variable and provided in the meta_output argument to save time. Counting rows in very large files can take some time.

Usage

```
bmeta(file = NULL, ...)
```

Arguments

file	String. Name or full path to a file compatible with data.table::fread()
...	Arguments that must be passed to data.table::fread() like "sep".

Value

A list of 2

Examples

```
file <- system.file("extdata", "test.csv", package = "bread")
## Filtering on 2 columns, using regex.
meta_output <- bmeta(file = file)
```

---

bnrow *Count the number of rows of a big file without loading it in memory*

Description

Counts the number of rows using data.table::fread() and the "wc" Unix command. This allows analyzing big files that would not fit in memory (and cause an error such as "cannot allocate vector of size").

Usage

```
bnrow(file = NULL)
```

Arguments

file	String. Name or full path to a file compatible with data.table::fread()

Value

A numeric

Examples

```
file <- system.file("extdata", "test.csv", package = "bread")
## Counting rows (almost like the band)
bnrow(file = file)
```

bread *Reads a file in table format, selecting columns, subsetting rows by number and filtering them by column values*

Description

Wrapper for data.table::fread() simplifying the use of Unix commands like grep, cut, awk and sed on a data file *before* loading it in memory. The Unix commands are automatically generated from the arguments. This is useful if you want to load a big file too large for your available memory (and encounter the "cannot allocate vector of size" error) and know you can work on a subsample. "b" stands for "big file". This function allows to subset rows by their index number, select columns and filter with a pattern.

Usage

```
bread(
  file = NULL,
  first_row = NULL,
  last_row = NULL,
  head = NULL,
  tail = NULL,
  colnames = NULL,
  colnums = NULL,
  patterns = NULL,
  filtered_columns = NULL,
  fixed = FALSE,
  ...
)
```

Arguments

file	String. Name or full path to a file compatible with data.table::fread()
first_row	Numeric. First row of the portion of the file to subset.
last_row	Numeric. Last row of the portion of the file to subset.
head	Numeric. How many rows starting from the first in the file.
tail	Numeric. How many rows starting from the last in the file.
colnames	Vector of strings. Exact names of columns to select. If both colnames and colnums are provided, colnums will be prefered.
colnums	Vector of numeric. Columns index numbers.
patterns	Vector of strings. One or several patterns used to filter the data from the input file. Each element of the vector should correspond to the column to be filtered. Can use regular expressions.
filtered_columns	Vector of strings or numeric. Optional. The columns to be filtered should be indicated through their names or their index number. Each element of the vector should correspond to the pattern with which it will be filtered.

fixed	Logical. If TRUE, pattern is a string to be matched as is. Overrides all conflicting arguments.
...	Arguments that must be passed to data.table::fread() like "sep" or "dec".

Details

You can mix and match the row subsetting, the filtering by value and the selecting of columns. In order, the function:

1. subsets the rows by their numbers (with sed & awk). You need to input the index number of the first and last rows you want to load in memory with fread(), or alternatively use either the head or tail arguments to subset the first or last rows of the file.
2. selects columns by index number or name (with cut). If both colnames and colnums are provided, colnums will be prefered.
3. filters the data selected so far with a pattern by column (with grep). The columns to be filtered should be indicated through their names or their index number. Each element of the vector should correspond to the pattern with which it will be filtered.#'

Value

A data frame with the selected columns and the subsetted and filtered data

Warning

Best practice would probably be to load the big file in a SQL database or something. Or not working on huge CSV files in the first place. But if you have to, you hopefully won't have to delve into the fascinating grammar of Unix commands.

Examples

```
file <- system.file("extdata", "test.csv", package = "bread")
## Select the columns numbered 1 and 3
bread(file = file, colnums = c(1,3))
## Select the columns named "YEAR" and "PRICE", then filter to keep only the
## value "2022" in column "YEAR"
bread(file = file, colnames = c("YEAR", "PRICE"),
      patterns = 2002, filtered_columns = "YEAR")
## Subset to keep only the rows 10 to 18, select the columns named "YEAR"
## and "COLOR" then filter to keep only the value "red" in column "COLOR"
bread(file = file, colnames = c("YEAR", "COLOR"),
      patterns = "red", filtered_columns = "COLOR",
      first_row = 10, last_row = 18)
```

bselect *Pre-selects columns of a data file before loading it in memory*

Description

Simple wrapper for data.table::fread() allowing to select columns of data from a file with the Unix cut command. This method is useful if you want to load a file too large for your available memory (and encounter the "cannot allocate vector of size" error).

Usage

```
bselect(file = NULL, colnames = NULL, colnums = NULL, ...)
```

Arguments

file	String. Full path to a file
colnames	Vector of strings. Exact names of columns to select. If both colnames and colnums are provided, colnums will be prefered.
colnums	Vector of numeric. Columns index numbers.
...	Arguments that must be passed to data.table::fread() like "sep" or "dec".

Value

A dataframe with the selected columns

Examples

```
file <- system.file("extdata", "test.csv", package = "bread")
## Select the columns numbered 1 and 3
bselect(file = file, colnums = c(1,3))
## Select the columns named "PRICE" and "COLOR"
bselect(file = file, colnames = c("PRICE", "COLOR"))
```

bsep *Tries to identify the separator / delimiter used in a table format file*

Description

The function reads the first row and tests the following common separators by default: ";" "\t" " " "|" ":" ","

Usage

```
bsep(file, ntries = 10, separators = c(";", "\t", " ", "|", ":", ","))
```

Arguments

file	String. Name or full path to a file compatible with data.table::fread()
ntries	Numeric. Number of rows to check for
separators	Vector of strings. Additional uncommon delimiter to check for

Value

A string

Examples

```
file <- system.file("extdata", "test.csv", package = "bread")
## Checking the delimiter on the first 12 rows, including headers
bsep(file = file, ntries = 12)
```

---

bsubset *Pre-subsets rows of a data file by index number before loading it in memory*

Description

Simple wrapper for data.table::fread() allowing to subset rows of data from a file with the Unix sed or awk commands. This method is useful if you want to load a file too large for your available memory (and encounter the "cannot allocate vector of size" error). You need to input the index number of the first and last rows you want to load in memory with fread(), or alternatively use either the head or tail arguments to subset the first or last rows of the file.

Usage

```
bsubset(
  file = NULL,
  head = NULL,
  tail = NULL,
  first_row = NULL,
  last_row = NULL,
  ...
)
```

Arguments

file	String. Full path to a file
head	Numeric. How many rows starting from the first in the file.
tail	Numeric. How many rows starting from the last in the file.
first_row	Numeric. First row of the portion of the file to subset.
last_row	Numeric. Last row of the portion of the file to subset.
...	Arguments that must be passed to data.table::fread() like "sep".

Value

A dataframe containing the subsetted rows

Examples

```
file <- system.file("extdata", "test.csv", package = "bread")
## Head or Tail... for the first n or last n rows
bsubset(file = file, head = 5)
## Subset from the middle of a file
bsubset(file = file, first_row = 5, last_row = 10)
## first_row defaults as 1 and last_row as the last row of the file
bsubset(file = file, first_row = 5)
bsubset(file = file, last_row = 10)
```

Index