Package 'csvread'

January 8, 2015

Title Fast Specialized CSV File Loader.

Version 1.2

Author Sergei Izrailev

Maintainer Sergei Izrailev <sizrailev@jabiruventures.com>

Description This package provides functions for loading large (10M+ lines) CSV and other delimited files, similar to read.csv, but typically faster and using less memory than the standard R loader. While not entirely general, it covers many common use cases when the types of columns in the CSV file are known in advance. In addition, the package provides a class 'int64', which represents 64-bit integers exactly when reading from a file. The latter is useful when working with 64-bit integer identifiers exported from databases. The CSV file loader supports common column types including 'integer', 'double', 'string', and 'int64', leaving further type transformations to the user.

URL http://github.com/jabiru/csvread

Depends R (>= 2.15),
methods

Enhances bit64

License Apache License (== 2.0)

Copyright Copyright (C) Collective, Inc. | file inst/COPYRIGHTS

LazyData true

R topics documented:

csvread *Fast Specialized CSV File Loader.*

Description

Package csvread contains a fast specialized CSV and other delimited file loader, and a basic 64-bit integer class to aid in reading 64-bit integer values.

Given a list of the column types, function csvread parses the CSV file and returns a data frame.

map.coltypes guesses the column types in the CSV file by reading the first nrows lines. The result can be passed to csvread as the coltypes argument.

Usage

```
csvread(file, coltypes, header, colnames = NULL, nrows = NULL,
  verbose = FALSE, delimiter = ",", na.strings = c("NA", "na", "NULL",
  "null", ""))

map.coltypes(file, header, nrows = 100, delimiter = ",")
```

Arguments

file	Path to the CSV file.
coltypes	A vector of column types, e.g., c("integer", "string"). The accepted types are "integer", "double", "string", "long" and "longhex". • integer - the column is parsed into an R integer type (32 bit) • double - the column is parsed into an R double type • string - the column is loaded as character type • long - the column is interpreted as the decimal representation of a 64-bit integer, stored as a double and assigned the int64 class. • longhex - the column is interpreted as the hex representation of a 64-bit integer, stored as a double and assigned the int64 class with an additional attribute base = 16L that is used for printing. • integer64 - same as long but produces a column of class integer64, which should be compatible with package bit64 (untested). • verbose - if TRUE, the function prints number of lines counted in the file. • delimiter - a single character delimiter, defalut is ",".
header	TRUE (default) or FALSE; indicates whether the file has a header and serves as the source of column names if colnames is not provided.
colnames	Optional column names for the resulting data frame. Overrides the header, if header is present. If NULL, then the column names are taken from the header, or, if there is no header, the column names are set to 'COL1', 'COL2', etc.
nrows	If NULL, the function first counts the lines in the file. This step can be avoided if the number of lines is known by providing a value to nrows. On the other hand, nrows can be used to read only the first lines of the CSV file.
verbose	If TRUE and nrows is NULL, the function prints number of lines counted in the file.
delimiter	A single character delimiter, defalut is ",".
na.strings	A vector of strings to be considered NA in the input file.

Details

csvread provides functionality for loading large (10M+ lines) CSV and other delimited files, similar to read.csv, but typically faster and using less memory than the standard R loader. While not entirely general, it covers many common use cases when the types of columns in the CSV file are known in advance. In addition, the package provides a class 'int64', which represents 64-bit integers exactly when reading from a file. The latter is useful when working with 64-bit integer identifiers exported from databases. The CSV file loader supports common column types including integer, double, string, and int64, leaving further type transformations to the user.

If number of columns, which is inferred from the number of provided coltypes, is greater than the actual number of columns, the extra columns are still created. If the number of columns is less than the actual number of columns in the file, the extra columns in the file are ignored. Commas included in double quotes will be considered part of the field, rather than a separator, but double quotes will NOT be stripped. Runaway double quotes will end at the end of the line.

See also int64 for information about dealing with 64-bit integers when loading data from CSV files.

Value

A data frame containing the data from the CSV file.

Maintainer

Sergei Izrailev

Copyright

License

URL

http://github.com/jabiru/csvread

Installation from github

```
devtools::install_github("jabiru/csvread")
```

Author(s)

Sergei Izrailev

See Also

int64

Examples

```
## Not run:
## Basic use case when column types are known and there's no missing data.

frm <- csvread("inst/10rows.csv",
coltypes = c("longhex", "string", "double", "integer", "long"),
header = FALSE)

frm
# COL1       COL2     COL3 COL4 COL5
# 1  11fb89c1558c792 2011-05-06 0.150001 4970 4977
# 2  11fb89c1558c792 2011-05-06 0.150001 4970 4987
# 3  11fb89c1558c792 2011-05-06 0.150001 5200 5528
# 4  11fb89c1558c792 2011-05-06 0.150001 4970 5004
# 5  11fb89c1558c792 2011-05-06 0.150001 4970 4980
# 6  11fb89c1558c792 2011-05-06 0.150001 4970 5020
# 7  11fb89c1558c792 2011-05-06 0.150001 4970 5048
# 8  11fb89c1558c792 2011-05-06 0.150001 4970 5035
# 9  11fb89c1558c792 2011-05-06 0.150001 4970 4971
# 10 11fb89c1558c792 2011-05-06 0.150001 4970 4973

typeof(frm$COL1)
# [1] "double"
class(frm$COL1)
# [1] "int64"

typeof(frm$COL5)
# [1] "double"
class(frm$COL5)
# [1] "int64"

#### Examples with missing data.

## The input file contains values "NA", "NA ", " NA ", "NULL", "na"
## and missing fields in various columns.

writeLines(scan("inst/10rows_na.csv", "character", sep = "\n"))
# Read 10 items
# 11fb89c1558c792,2011-05-06,0.150001,4970,4977
# 11fb89c1558c792,2011-05-06,0.150001,4970,4987
# 11fb89c1558c792, NA ,0.150001,NA ,5528
# NA,2011-05-06,0.150001,4970,5004
# 11fb89c1558c792,na,0.150001,4970,4980
# 11fb89c1558c792,2011-05-06,NA,4970,5020
# 11fb89c1558c792,2011-05-06,0.150001,NULL,5048
# 11fb89c1558c792,2011-05-06,0.150001,4970,NA
# ,2011-05-06,0.150001,4970,4971
# 11fb89c1558c792,2011-05-06,0.150001,4970,

## By default, all missing fields in this input are handled, except
## for the " NA " in a character column COL3, which remains unchanged.
## This is the intended behavior, similar to that of read.csv.

frm <- csvread("inst/10rows_na.csv",
coltypes = c("longhex", "string", "double", "integer", "long"),
header = FALSE)
```

```
frm
# COL1       COL2     COL3 COL4 COL5
# 1  11fb89c1558c792 2011-05-06 0.150001 4970 4977
# 2  11fb89c1558c792 2011-05-06 0.150001 4970 4987
# 3  11fb89c1558c792        NA  0.150001   NA 5528
# 4             <NA> 2011-05-06 0.150001 4970 5004
# 5  11fb89c1558c792       <NA> 0.150001 4970 4980
# 6  11fb89c1558c792 2011-05-06       NA 4970 5020
# 7  11fb89c1558c792 2011-05-06 0.150001   NA 5048
# 8  11fb89c1558c792 2011-05-06 0.150001 4970 <NA>
# 9             <NA> 2011-05-06 0.150001 4970 4971
# 10 11fb89c1558c792 2011-05-06 0.150001 4970 <NA>

## End(Not run)
## Not run:
#### The column types can be guessed by using map.coltypes.

coltypes <- map.coltypes("inst/10rows.csv", header = FALSE)
coltypes
#       V1        V2        V3        V4        V5
# "string"  "string"  "double" "integer" "integer"

## Note the difference when "NA"s are present in an integer column 4,
## which is then considered to be a string column.
coltypes.na <- map.coltypes("inst/10rows_na.csv", header = FALSE)
coltypes.na
#        V1        V2        V3        V4        V5
#  "string"  "string"  "double"  "string" "integer"

frm <- csvread(file = "inst/10rows.csv", coltypes = coltypes,
   header = F, verbose = T)
# Counted 10 lines.

frm
#               COL1       COL2     COL3 COL4 COL5
# 1  11fb89c1558c792 2011-05-06 0.150001 4970 4977
# 2  11fb89c1558c792 2011-05-06 0.150001 4970 4987
# 3  11fb89c1558c792 2011-05-06 0.150001 5200 5528
# 4  11fb89c1558c792 2011-05-06 0.150001 4970 5004
# 5  11fb89c1558c792 2011-05-06 0.150001 4970 4980
# 6  11fb89c1558c792 2011-05-06 0.150001 4970 5020
# 7  11fb89c1558c792 2011-05-06 0.150001 4970 5048
# 8  11fb89c1558c792 2011-05-06 0.150001 4970 5035
# 9  11fb89c1558c792 2011-05-06 0.150001 4970 4971
# 10 11fb89c1558c792 2011-05-06 0.150001 4970 4973
typeof(frm$COL1)
# [1] "character"
class(frm$COL1)
# [1] "character"

typeof(frm$COL5)
# [1] "integer"
class(frm$COL5)
# [1] "integer"

## Convert the first column to int64 manually
```

```
frm$COL1 <- as.int64(frm$COL1, base = 16)
frm$COL1
# [1] "11fb89c1558c792" "11fb89c1558c792" "11fb89c1558c792" "11fb89c1558c792"
# [5] "11fb89c1558c792" "11fb89c1558c792" "11fb89c1558c792" "11fb89c1558c792"
# [9] "11fb89c1558c792" "11fb89c1558c792"
typeof(frm$COL1)
# [1] "double"
class(frm$COL1)
# [1] "int64"

## Print the first value in base 10.
as.character.int64(frm$COL1[1], base = 10)
# [1] "80986298828507026"

#### Character (string) columns with NAs and non-default na.strings

## A file with NAs and missing values: note that the in the first
## column, an empty string in row 9 is not considered NA because
## na.strings are set to "NA". By default, the empty string will be
## considered NA. Also, in column 2, rows 3 and 5, the values are
## " NA " (with spaces) and "na", respectively, because they don't
## match values in na.strings and therefore are not considered to be NA.

coltypes
#       V1        V2        V3        V4        V5
# "string"  "string"  "double" "integer" "integer"

frm <- csvread(file = "inst/10rows_na.csv", coltypes = coltypes,
   header = F, verbose = T, na.strings = "NA")
# Counted 10 lines.

frm
#               COL1       COL2     COL3 COL4 COL5
# 1  11fb89c1558c792 2011-05-06 0.150001 4970 4977
# 2  11fb89c1558c792 2011-05-06 0.150001 4970 4987
# 3  11fb89c1558c792         NA 0.150001   NA 5528
# 4              <NA> 2011-05-06 0.150001 4970 5004
# 5  11fb89c1558c792         na 0.150001 4970 4980
# 6  11fb89c1558c792 2011-05-06       NA 4970 5020
# 7  11fb89c1558c792 2011-05-06 0.150001   NA 5048
# 8  11fb89c1558c792 2011-05-06 0.150001 4970   NA
# 9                  2011-05-06 0.150001 4970 4971
# 10 11fb89c1558c792 2011-05-06 0.150001 4970   NA


## End(Not run)
```

`int64` *A very basic 64-bit integer class.*

Description

A very basic 64-bit integer class.

Usage

```
int64(length = 0)

is.int64(x)

## Default S3 method:
as.int64(x, ...)

## S3 method for class 'factor'
as.int64(x, ...)

## S3 method for class 'character'
as.int64(x, base = 10L, ...)

## S3 method for class 'numeric'
as.int64(x, ...)

## S3 method for class 'NULL'
as.int64(x, ...)

## S3 method for class 'int64'
format(x, ...)

## S3 method for class 'int64'
print(x, ...)

## S3 method for class 'int64'
as.character(x, base = NULL, ...)

## S3 method for class 'int64'
as.double(x, ...)

## S3 method for class 'int64'
as.integer(x, ...)

## S3 method for class 'int64'
is.na(x, ...)

## S3 method for class 'int64'
as.data.frame(x, ...)

## S3 method for class 'int64'
as.list(x, ...)

## S3 method for class 'int64'
c(...)

## S3 method for class 'int64'
is.numeric(x)

## S3 method for class 'int64'
rep(x, ...)
```

Arguments

length	A non-negative integer specifying the desired length. Double values will be coerced to integer: supplying an argument of length other than one is an error.
x	Object to be coerced or tested
...	Further arguments passed to or from other methods.
base	Specifies the base of the number (default is the base attribute of the object).

Details

The int64 class stores 64-bit integers in vectors of doubles and the base as an attribute base of the vector for printing and conversion to character. The motivation behind this class is to give R the ability to load 64-bit integers directly, for example, to represent the commonly used 64-bit identifiers in relational and other databases.

See Also

Ops.int64 csvread

Ops.int64 *Operators for the* int64 *class.*

Description

Operators for the int64 class: one of +, −, ==, !=, <, <=, > or >=.

Usage

```
e1 + e2
e1 - e2

## S3 method for class 'int64'
e1 + e2

## S3 method for class 'int64'
e1 - e2
```

Arguments

e1	int64 object, character vector or numeric vector (character and numeric values are converted by as.int64).
e2	int64 object, character vector or numeric vector (character and numeric values are converted by as.int64).

See Also

int64

Index