Package 'readsparse'

October 14, 2021

Type Package

Title Read and Write Sparse Matrices in 'SVMLight' and 'LibSVM' Formats

Version 0.1.5-1

Author David Cortes

Maintainer David Cortes <david.cortes.rivera@gmail.com>

URL https://github.com/david-cortes/readsparse

BugReports https://github.com/david-cortes/readsparse/issues

Description Read and write labelled sparse matrices in text format as used by software such as 'SVMLight', 'LibSVM', 'ThunderSVM', 'LibFM', 'xLearn', 'XGBoost', 'LightGBM', and others. Supports labelled data for regression, classification (binary, multi-class, multi-label), and ranking (with 'qid' field), and can handle header metadata and comments in files.

License BSD_2_clause + file LICENSE

Imports Rcpp (>= 1.0.5), Matrix, methods

Suggests MatrixExtra, testthat

LinkingTo Rcpp

RoxygenNote 7.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-10-14 04:30:04 UTC

R topics documented:

`read.sparse` *Read Sparse Matrix from Text File*

Description

Read a labelled sparse CSR matrix in text format as used by libraries such as SVMLight, LibSVM, ThunderSVM, LibFM, xLearn, XGBoost, LightGBM, and more.

The format is as follows:

`<label(s)> <column>:<value> <column>:<value> ...`

with one line per observation/row.

Example line (row):

`1 1:1.234 3:20`

This line denotes a row with label (target variable) equal to 1, a value for the first column of 1.234, a value of zero for the second column (which is missing), and a value of 20 for the third column.

The labels might be decimal (for regression), and each row might contain more than one label (must be integers in this case), separated by commas **without** spaces inbetween - e.g.:

`1,5,10 1:1.234 3:20`

This line indicates a row with labels 1, 5, and 10 (for multi-class classification). If the line has no labels, it should still include a space before the features.

The rows might additionally contain a 'qid' parameter as used in ranking algorithms, which should always lay inbetween the labels and the features and must be an integer - e.g.:

`1 qid:2 1:1.234 3:20`

The file might optionally contain a header as the first line with metadata (number of rows, number of columns, number of classes). Presence of a header will be automatically detected, and is recommended to include it for speed purposes. Datasets from the extreme classification repository (see references) usually include such a header.

Lines might include comments, which start after a '#' character. Lines consisting of only a '#' will be ignored. When reading from a file, such file might have a BOM (information about encoding uses in Windows sytems), which will be automatically skipped.

Usage

```
read.sparse(
  file,
  multilabel = FALSE,
  has_qid = FALSE,
  integer_labels = FALSE,
  index1 = TRUE,
  sort_indices = TRUE,
  ignore_zeros = TRUE,
  min_cols = 0L,
  min_classes = 0L,
```

```
  limit_nrows = 0L,
  no_trailing_ws = FALSE,
  from_string = FALSE
)
```

Arguments

file	Either a file path from which the data will be read, or a string ('character' variable) containing the text from which the data will be read. In the latter case, must pass 'from_string=TRUE'.
multilabel	Whether the input file can have multiple labels per observation. If passing 'multilabel=FALSE' and it turns out to have multiple labels, will only take the first one for each row. If the labels are non-integers or have decimal point, the results will be invalid.
has_qid	Whether the input file has 'qid' field (used for ranking). If passing 'FALSE' and the file does turns out to have 'qid', the features will not be read for any observations.
integer_labels	Whether to output the observation labels as integers.
index1	Whether the input file uses numeration starting at 1 for the column numbers (and for the label numbers when passing 'multilabel=TRUE'). This is usually the case for files downloaded from the repositories in the references. The function will check for whether any of the column indices is zero, and will ignore this option if so (i.e. will assume it is 'FALSE').
sort_indices	Whether to sort the indices of the columns after reading the data. These should already be sorted in the files from the repositories in the references.
ignore_zeros	Whether to avoid adding features which have a value of zero. If the zeros are caused due to numerical rounding in the software that wrote the input file, they can be post-processed by passing 'ignore_zeros=FALSE' and then something like 'X@x[X@x == 0] = 1e-8'.
min_cols	Minimum number of columns that the output 'X' object should have, in case some columns are all missing in the input data.
min_classes	Minimum number of columns that the output 'y' object should have, in case some columns are all missing in the input data. Only used when passing 'multilabel=TRUE'.
limit_nrows	Maximum number of rows to read from the data. If there are more than this number of rows, it will only read the first 'limit_nrows' rows. If passing zero (the default), there will be no row limit.
no_trailing_ws	Whether to assume that lines in the file will never have extra whitespaces
from_string	Whether to read the data from a string variable instead of a file. If passing 'from_string=TRUE', then 'file' is assumed to be a variable with the data contents on it.

Details

Note that this function:

- Will not make any checks for negative column indices.
- Has a precision of C type ‘int‘ for column indices and integer labels (the maximum value that this type can hold can be checked in ‘.Machine$integer.max‘).
- Will fill missing labels with NAs when passing ‘multilabel=FALSE‘.
- Will fill with zeros (empty values) the lines that are empty (that is, they generate a row in the data), but will ignore (that is, will not generate a row in the data) the lines that start with ‘#‘.

Be aware that the data is represented as a CSR matrix with index pointer of class C ‘int‘, thus the number of rows/columns/non-zero-elements cannot exceed ‘.Machine$integer.max‘.

On Windows, if the package is installed from CRAN and compiled using the GCC compiler version 4 or earlier (the default in older versions of RTools, such as Rtools35), it will not be able to read from or write to file names with non-ASCII characters, which can be solved by installing it directly from the GitHub repository (‘remotes::install_github("david-cortes/readsparse")‘). Whether support for non-ASCII file names is available or not can be checked through readsparse_nonascii_support.

On 64-bit Windows systems, if compiling the library with a compiler other than MinGW or MSVC, it will not be able to read files larger than 2GB. This should not be a concern if installing it from CRAN or from R itself, as the Windows version at the time of writing can only be compiled with MinGW.

If the file contains a header, and this header denotes a larger number of columns or of labels than the largest index in the data, the resulting object will have this dimension set according to the header. The third entry in the header (number of classes/labels) will be ignored when passing ‘multilabel=FALSE‘.

The function uses different code paths when reading from a file or from a string, and there might be slight differences between the obtained results from them. For example, reading from a file might produce the desired output if the file uses tabs as separators instead of spaces (not supported by most other software and not standard), whereas reading from a string will not. If any such difference is encountered, please submit a bug report in the package's GitHub page.

Value

A list with the following entries:

- ‘X‘: the features, as a CSR Matrix from package ‘Matrix‘ (class ‘dgRMatrix‘).
- ‘y‘: the labels. If passing ‘multilabel=FALSE‘ (the default), will be a vector (class ‘numeric‘ when passing ‘integer_labels=FALSE‘, class ‘integer‘ when passing ‘integer_labels=TRUE‘), otherwise will be a binary CSR Matrix (class ‘ngRMatrix‘).
- ‘qid‘: the query IDs used for ranking, as an integer vector. This entry will **only** be present when passing ‘has_qid=TRUE‘.

These can be easily transformed to other sparse matrix types through e.g. ‘X <- as(X, "CsparseMatrix")‘.

References

Datasets in this format can be found here:

- LibSVM Data: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

- Extreme Classification Repository: http://manikvarma.org/downloads/XC/XMLRepository.html

The format is also described at the SVMLight webpage: http://svmlight.joachims.org.

See Also

write.sparse

Examples

```
library(Matrix)
library(readsparse)

### Example input file
"1 2:1.21 5:2.05
-1 1:0.45 3:0.001 4:-10" -> coded.matrix

r <- read.sparse(coded.matrix, from_string=TRUE)
print(r)

### Convert it back to text
recoded.matrix <- write.sparse(file=NULL, X=r$X, y=r$y, to_string=TRUE)
cat(recoded.matrix)

### Example with real file I/O
## generate a random sparse matrix and labels
set.seed(1)
X <- rsparsematrix(nrow=5, ncol=10, nnz=8)
y <- rnorm(5)

## save into a text file
temp_file <- file.path(tempdir(), "matrix.txt")
write.sparse(temp_file, X, y, integer_labels=FALSE)

## inspect the text file
cat(paste(readLines(temp_file), collapse="\n"))

## read it back
r <- read.sparse(temp_file)
print(r)

### (Note that columns with all-zeros are discarded,
###  this behavior can be avoided with 'add_header=TRUE')
```

---

readsparse_nonascii_support

Determine if readsparse will handle non-ASCII file paths

Description

See read.sparse for details. In general, the library will be able to handle non-ASCII file paths, unless it is compiled with G++ version 4 or earlier (the default under Windows for older versions of RTools, such as RTools35).

Usage

```
readsparse_nonascii_support()
```

Value

A logical/boolean value telling whether non-ASCII file names will be supported or not.

`write.sparse` *Write Sparse Matrix in Text Format*

Description

Write a labelled sparse matrix into text format as used by software such as SVMLight, LibSVM, ThunderSVM, LibFM, xLearn, XGBoost, LightGBM, and others - i.e.:

`<labels(s)> <column:value> <column:value> ...`

For more information about the format and usage examples, see read.sparse.

Can write labels for regression, classification (binary, multi-class, and multi-label), and ranking (with 'qid'), but note that most software that makes use of this data format supports only regression and binary classification.

Usage

```
write.sparse(
  file,
  X,
  y,
  qid = NULL,
  integer_labels = TRUE,
  index1 = TRUE,
  sort_indices = TRUE,
  ignore_zeros = TRUE,
  add_header = FALSE,
  decimal_places = 8L,
  append = FALSE,
  to_string = FALSE
)
```

Arguments

file	Output file path into which to write the data. Will be ignored when passing 'to_string=TRUE'.
X	Sparse data to write. Can be a sparse matrix from package 'Matrix' (classes: 'dgRMatrix', 'dgTMatrix', 'dgCMatrix', 'ngRMatrix', 'ngTMatrix', 'ngCMatrix') or from package 'SparseM' (classes: 'matrix.csr', 'matrix.coo', 'matrix.csc'), or a dense matrix of all numeric values, passed either as a 'matrix' or as a 'data.frame'. If 'X' is a vector (classes 'numeric', 'integer', 'dsparseVector'), will be assumed to be a row vector and will thus write one row only. Note that the data will be casted to 'dgRMatrix' in any case.
y	Labels for the data. Can be passed as a vector ('integer' or 'numeric') if each observation has one label, or as a sparse or dense matrix (same format as 'X') if each observation can have more than 1 label. In the latter case, only the non-missing column indices will be written, while the values are ignored.
qid	Secondary label information used for ranking algorithms. Must be an integer vector if passed. Note that not all software supports this.
integer_labels	Whether to write the labels as integers. If passing 'FALSE', they will have a decimal point regardless of whether they are integers or not. If the file is meant to be used for a classification algorithm, one should pass 'TRUE' here (the default). For multilabel classification, the labels will always be written as integers.
index1	Whether the column and label indices (if multi-label) should have numeration starting at 1. Most software assumes this is 'TRUE'.
sort_indices	Whether to sort the indices of 'X' (and of 'y' if multi-label) before writing the data. Note that this will cause in-place modifications if either 'X' or 'y' are passed as CSR matrices from the 'Matrix' package.
ignore_zeros	Whether to ignore (not write) features with a value of zero after rounding to the specified decimal places.
add_header	Whether to add a header with metadata as the first line (number of rows, number of columns, number of classes). If passing 'integer_label=FALSE' and 'y' is a vector, will write zero as the number of labels. This is not supported by most software.
decimal_places	Number of decimal places to use for numeric values. All values will have exactly this number of places after the decimal point. Be aware that values are rounded and might turn to zeros (will be skipped by default) if they are too small (one can do something like 'X@x <- ifelse(X@x >= 0, pmin(X@x, 1e-8), pmax(X@x, -1e-8))' to avoid this).
append	Whether to append text at the end of the file instead of overwriting or creating a new file. Ignored when passing 'to_string=TRUE'.
to_string	Whether to write the result into a string (which will be returned from the function) instead of into a file.

Details

Be aware that writing sparse matrices to text is not a lossless operation - that is, some information might be lost due to numeric precision, and metadata such as row and column names will not be saved. It is recommended to use ‘saveRDS‘ or similar for saving data between R sessions, or to use binary formats for passing between different software such as R->Python.

The option ‘ignore_zeros‘ is implemented heuristically, by comparing ‘abs(x) >= 10^(-decimal_places)/2‘, which might not match exactly with the rounding that is done implicitly in string conversions in the libc/libc++ functions - thus there might still be some corner cases of all-zeros written into features if the (absolute) values are very close to the rounding threshold.

While R uses C ‘double‘ type for numeric values, most of the software that is able to take input data in this format uses ‘float‘ type, which has less precision.

The function uses different code paths when writing to a file or to a string, and there might be slight differences between the generated texts from them. If any such difference is encountered, please submit a bug report in the package's GitHub page.

Value

If passing ‘to_string=FALSE‘ (the default), will not return anything (‘invisible(NULL)‘). If passing ‘to_string=TRUE‘, will return a ‘character‘ variable with the data contents written into it.

See Also

read.sparse

Index