CSV Reader/Writer

Generated by Doxygen 1.9.1

Chapter 1

A very tiny simple CSV file reader/writer for C++

Not for a professional use, but for just visualize, manipulate and print your comma separated data.

1.1 Usage:

First you need to instanciate a reader/writer, suppose you have a `float` file to read (double and int dtypes are supported):

```
CSVRW<float>* rw = new CSVRW<float>(dtype::F)
```

`dtype::F` is an `enum` variable defined in `CSVRW.h` and it needs to manual provide the data type for a better data fetching and memory management (no more needed in C++20).

Once instanciated a reader/writer, a `CSVFile` variable needs to be constructed.
Two examples are needed here.
Suppose we have data to read stored in a local csv file:

`CSVFile<float>* myfile = new CSVFile<float>()` --> *empty in this case, it will work as a to-fill-container taking data from our file*

So now we can call our reader/writer ready to read:

‘rw->read_file("path/to/file", myfile, true, ’,')` --> *the reader/writer will fill our container*

Now, `myfile` contains your file data.

You can access to your data typing:

```
myfile->getHeader() // Retrieve columns names as string vector
myfile->getData() //And you can access like a matrix by [][] operator
```

Second Example: you have your data stored in a matrix of `float`:

```
float my_float_data[ROWS][COLS]; // Suppose this is the filled matrix
vector<string> col_names = {"col1","col2",..."col3"};
CSVFile<float>* my_csv_file = new CSVFile<float>(&col_names, &my_float_↩
data, ROWS, COLS)
```

Once built the `CSVFile<float>` object you can, for example manipulate your data and then write out to a file:

```
rw->write_file("/path/to/file.csv",my_csv_file,',');
```

A message will confirm that output is completed.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 CSVFile< T > Class Template Reference

`#include <CSVFile.h>`

Public Member Functions

- CSVFile ()

  Construct an empty CSVFile class object.
- CSVFile (std::vector< std::string > header, T **data, int rows, int cols)

  Construct a CSVFile starting from local variables.
- void head (int heads=5)

  Print elements to stdout (default is 5 elements)
- std::vector< std::string > **getHeader** ()
- std::vector< std::vector< T > > **getData** ()
- void **appendToHeader** (std::string element)
- void **appendRowToData** (std::vector< T > row)
- int getDataRows ()

  Safe getter of number of rows.
- int getDataCols ()

  Safe getter of number of cols.

3.1.1 Detailed Description

template<class T>
class CSVFile< T >

This is CSVFile class, use this calss for manage or create your CSV file. This class has two attributes:

- `header` (`vector<string>`) : Usually a CSV File contains an header for Columns Names
- data (`vector<vector<T>>`): The data (numerical) contained in the file.

Constructor:

- No params: Creates empty CSVFile Object

Constructor with Parameters:

- `vector<string>` header: if you want to build CSVFile object starting from aleady existing data
- `T**` data : if you have a matrix of numerical data

Methods:

- appendToHeader: Append element to header (kept private)

- appendRowToData: Append a vector of values to data matrix (kept private)
- getters

3.1.2 Constructor & Destructor Documentation

3.1.2.1 CSVFile()

```
template<class T >
CSVFile< T >::CSVFile (
            std::vector< std::string > header,
            T ** data,
            int rows,
            int cols )
```

Construct a CSVFile starting from local variables.

Parameters

(vector<string>)*	header: string vector containing all columns names
*(T**)*	data: variable containing numerical data e.g. a float matrix
(int)	rows: data matrix total number of rows
(int)	cols: data matrix total number of cols

3.1.3 Member Function Documentation

3.1.3.1 head()

```
template<class T >
void CSVFile< T >::head (
            int heads = 5 )
```

Print elements to stdout (default is 5 elements)

Parameters

(int)	heads: number of elements to print

The documentation for this class was generated from the following file:

- CSVFile.h

3.2 CSVRW< T > Class Template Reference

This is the CSVRW (CSV Read & Write) This class can help to read and write CSV files just instanciating this object.
`#include <CSVRW.h>`

Public Member Functions

- **CSVRW** (dtypes dt)
- void read_file (std::string filepath, CSVFile< T > *file, bool header=true, char delim=',')

  Read a csv file.
- void write_file (std::string filename, CSVFile< T > *file, char delim=',')

Write a csv file.

3.2.1 Detailed Description

template<class T>
class CSVRW< T >

This is the CSVRW (CSV Read & Write) This class can help to read and write CSV files just instanciating this object. So this class needs to be instanciated by using this dtype flags. Manual data type control can improve memory space allocation.

Attention

This class can only read a file like:

|col1 |col2 |col3 |...|col_m | -->header
|val11|val12|val13|...|val_1m| -->data (numeric)
[...]
|valn1|valn2|valn3|...|val_nm|

Remarks

Columns having categorical data types are not supported, consider to encode your data to numeric.

Author

Jacopo Vitale MSc. - Computer Science Engineering University of Cassino and Southern Latium

Date

Feb 23 2023

3.2.2 Member Function Documentation

3.2.2.1 read_file()

```
template<class T >
void CSVRW< T >::read_file (
            std::string filepath,
            CSVFile< T > * file,
            bool header = true,
            char delim = ',' )
```

Read a csv file.

Parameters

(string)	filepath: path/to/file
(CSVFile∗)	file: variable containing csv fetched data
(bool)	header: if the file contains an header (columns names)

3.2.2.2 write_file()

```
template<class T >
void CSVRW< T >::write_file (
            std::string filename,
```

```
CSVFile< T > * file,
char delim = ',' )
```

Write a csv file.

Parameters

(string)	filepath: path/to/write (including .csv)
(CSVFile∗)	file: variable containing csv data to write
(char)	delim: custom delimiter default is comma

The documentation for this class was generated from the following file:

- CSVRW.h

Index