Data Frame

0.1

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 DF Namespace Reference

Classes

- class DataFrame

  DataFrame is class for parsing data in a given file with a give delimeter (default is comma ','). all data will be saved as string wich user can later covert to desired type.

4.2 std Namespace Reference

Namespaces

- shorts

  namespace for introducting shortnames

4.3 std::shorts Namespace Reference

namespace for introducting shortnames

Typedefs

- using V_string = vector< string >
- using VV_string = vector< V_string >
- using V_any = vector< any >
- using VV_any = vector< vector< any > >
- using Data = unordered_map< string, V_string >
- using V_double = vector< double >
- using V_int = vector< int >
- using V_pair_ints = std::vector< std::pair< int, int > >

4.3.1 Detailed Description

namespace for introducting shortnames

4.3.2 Typedef Documentation

4.3.2.1 Data

```
using std::shorts::Data = typedef unordered_map<string, V_string>
```

Definition at line 39 of file ReadFiles.hpp.

4.3.2.2 V_any

```
using std::shorts::V_any = typedef vector<any>
```

Definition at line 37 of file ReadFiles.hpp.

4.3.2.3 V_double

```
using std::shorts::V_double = typedef vector<double>
```

Definition at line 40 of file ReadFiles.hpp.

4.3.2.4 V_int

```
using std::shorts::V_int = typedef vector<int>
```

Definition at line 41 of file ReadFiles.hpp.

4.3.2.5 V_pair_ints

```
using std::shorts::V_pair_ints = typedef std::vector<std::pair<int, int> >
```

Definition at line 42 of file ReadFiles.hpp.

4.3.2.6 V_string

```
using std::shorts::V_string = typedef vector<string>
```

Definition at line 35 of file ReadFiles.hpp.

4.3.2.7 VV_any

```
using std::shorts::VV_any = typedef vector<vector<any> >
```

Definition at line 38 of file ReadFiles.hpp.

4.3.2.8 VV_string

```
using std::shorts::VV_string = typedef vector<V_string>
```

Definition at line 36 of file ReadFiles.hpp.

Chapter 5

Class Documentation

5.1 DF::DataFrame Class Reference

DataFrame is class for parsing data in a given file with a give delimeter (default is comma ','). all data will be saved as string wich user can later covert to desired type.

`#include <ReadFiles.hpp>`

Public Member Functions

- int get_n_rows () const

  Get the number of rows of a given data.
- int get_n_cols () const

  Get the number cols of a given data.
- void set_headers (std::shorts::V_string const &v_hdrs)

  Set the headers with user provided vector of strings.
- std::shorts::V_string get_headers () const

  Get the headers.
- std::shorts::V_string get_by_header (std::string const &hdr)
- DataFrame copy () const

  to copy current data into new data frame
- DataFrame copy_by_headers (std::shorts::V_string const &v_hdrs)

  copy the requested data by provided headers name as vector of strings into a new data frame
- void remove_duplications (std::string const &hdr)

  get a given header and only keep the first occurance and remove the remaning rows
- void read_files (std::string_view path, char delim=',', bool is_first_col_header=true, std::shorts::V_string v_↩ hdrs={})

  read files
- void read_text (std::string const &text, std::shorts::V_pair_ints const &v_cols_start_length, bool is_first_col↩ _header=true, std::shorts::V_string v_hdrs={})
- void head (unsigned long long n=5)

  print n first rows off all columns
- void swap_cols_pos (std::string first_hdr, std::string second_hdr)

  swap two columns with their header with each others
- void add_col (std::shorts::V_string const &v_values, std::string hdr="new_col")

  add a column to the end of the dataframe based on a provided vector of string

- void add_col_of (std::string const &value, std::string hdr="new_col")

  dd a column to the end of the dataframe based on a provided value for all rows
- void append (std::vector< DataFrame > &&v_dfs)

  check if headers are the same then append the values
- void clear ()

  to clear headers and data
- std::shorts::V_string & operator[ ] (std::string hdr)

  subscript operator
- void write (std::string_view path, char delimiter=',')

  write the dataframe in a file with given path and delimiter
- void save_as_csv (std::string_view path)

  saving dataframe in comma separted format file

Public Attributes

- unsigned long long n_rows

  number of rows
- unsigned long long n_cols

  number of columns
- std::unordered_map< int, int > mising_values

  an unorderd maps for missing values

Private Member Functions

- std::shorts::V_string read_lines (std::string_view path)
- std::shorts::V_string read_lines (std::string const &text)
- std::shorts::V_string parse_line (std::string const &line, char delim)
- std::shorts::V_string parse_line (std::string const &line, std::shorts::V_pair_ints const &v_cols_start_ends)
- void fill_data (std::shorts::V_string const &v_lines, char delim=',', bool is_first_col_header=true, std::shorts::V_string v_hdrs={})
- void fill_data (std::shorts::V_string const &v_lines, std::shorts::V_pair_ints const &v_cols_start_length, bool is_first_col_header=true, std::shorts::V_string v_hdrs={})
- void insert_col (std::shorts::V_string const &values, std::string hdr)

Private Attributes

- std::shorts::Data data
- std::shorts::V_string headers

5.1.1 Detailed Description

DataFrame is class for parsing data in a given file with a give delimeter (default is comma ','). all data will be saved as string wich user can later covert to desired type.

Definition at line 54 of file ReadFiles.hpp.

5.1.2 Member Function Documentation

5.1.2.1 add_col()

```
void DF::DataFrame::add_col (
            std::shorts::V_string const & v_values,
            std::string hdr = "new_col" )
```

add a column to the end of the dataframe based on a provided vector of string

Parameters

v_strs	vector of values to add
hdr	given header name (default new: new_col), if the header is ther it would modified the header name

Definition at line 407 of file ReadFiles.cpp.

```
409 {
410     insert_col(values, hdr);
411 }
```

5.1.2.2 add_col_of()

```
void DF::DataFrame::add_col_of (
            std::string const & value,
            std::string hdr = "new_col" )
```

dd a column to the end of the dataframe based on a provided value for all rows

Parameters

value	value to be add to all rows
hdr	given header name (default new: new_col), if the header is ther it would modified the header name

Definition at line 413 of file ReadFiles.cpp.

```
414 {
415     std::shorts::V_string values(data[headers[0]].size(), value);
416     insert_col(values, hdr);
417 }
```

5.1.2.3 append()

```
void DF::DataFrame::append (
            std::vector< DataFrame > && v_dfs )
```

check if headers are the same then append the values

Parameters

v_dfs	

Definition at line 425 of file ReadFiles.cpp.

```
426 {
427     if(v_dfs.empty()) return;
428
429     for(auto& curr_df : v_dfs)
430     {
431         if(headers == curr_df.get_headers())
432         {
433             for(auto curr_hdr : headers)
434             {
435                 data[curr_hdr].insert(data[curr_hdr].end(), curr_df.data.at(curr_hdr).begin(),
      curr_df.data.at(curr_hdr).end());
436             }
437         }
438         curr_df.clear();
439     }
440     n_rows = data[headers[0]].size();
441     n_cols = headers.size();
442 }
```

5.1.2.4 clear()

```
void DF::DataFrame::clear ( )
```

to clear headers and data

Definition at line 419 of file ReadFiles.cpp.

```
420 {
421     headers.clear();
422     data.clear();
423 }
```

5.1.2.5 copy()

```
DataFrame DF::DataFrame::copy ( ) const
```

to copy current data into new data frame

Returns

DataFrame new data frame

5.1.2.6 copy_by_headers()

```
DF::DataFrame DF::DataFrame::copy_by_headers (
            std::shorts::V_string const & v_hdrs )
```

copy the requested data by provided headers name as vector of strings into a new data frame

Parameters

v_hdrs	

Returns

DataFrame new data frame

Definition at line 365 of file ReadFiles.cpp.

```
366 {
367     DF::DataFrame new_df;
368     new_df.n_cols = v_hdrs.size();
369     new_df.n_rows = n_rows;
370     new_df.headers = v_hdrs;
371 
372     for(auto const& hdr : v_hdrs)
373     {
374         new_df.data[hdr] = data[hdr];
375     }
376 
377     return new_df;
378 }
```

5.1.2.7 fill_data() [1/2]

```
void DF::DataFrame::fill_data (
              std::shorts::V_string const & v_lines,
              char delim = ',',
              bool is_first_col_header = true,
              std::shorts::V_string v_hdrs = {} )  [private]
```

Definition at line 106 of file ReadFiles.cpp.

```
107 {
108     std::shorts::VV_string vv_strs;
109 
110     for(auto const& line : lines)
111     {
112         std::shorts::V_string v_str_tmp = parse_line(line, delim);
113         vv_strs.emplace_back(v_str_tmp);
114     }
115 
116     // init n_rows and n_cols
117     n_rows = vv_strs.size();
118     n_cols = vv_strs[0].size();
119 
120     headers.resize(n_cols);
121 
122     // initializing headers
123     for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
124     {
125         if(is_first_col_header)
126         {
127             headers[i_col] = vv_strs[0][i_col];
128         }
129         else if(v_hdrs.size() > 0)
130         {
131             if(v_hdrs.size() == n_cols)
132             {
133                 headers[i_col] = v_hdrs[i_col];
134             }
135             else
136             {
137                 throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: number of provided
      headers does not match with the number of columns in the data"));
138             }
139         }
140         else
141         {
142             headers[i_col] = std::to_string(i_col + 1);
143         }
```

```
144     }
145
146     for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
147     {
148         std::shorts::V_string values;
149         for(unsigned long long i_row{is_first_col_header}; i_row < n_rows; ++i_row)
150         {
151             if(vv_strs[i_row].size() != n_cols)
152             {
153                 throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: inconsistent number of
    columns, check row {}", i_row + 1));
154             }
155
156             values.emplace_back(vv_strs[i_row][i_col]);
157
158             // check for the missig values
159             // missing values are empty string, NA, and NAN
160             if(vv_strs[i_row][i_col].empty() ||
161                vv_strs[i_row][i_col] == "NA" ||
162                vv_strs[i_row][i_col] == "NAN")
163             {
164                 mising_values.insert({i_row, i_col});
165             }
166         }
167
168         data[headers[i_col]] = values;
169     }
170 }
```

5.1.2.8 fill_data() [2/2]

```
void DF::DataFrame::fill_data (
            std::shorts::V_string const & v_lines,
            std::shorts::V_pair_ints const & v_cols_start_length,
            bool is_first_col_header = true,
            std::shorts::V_string v_hdrs = {} )  [private]
```

Definition at line 172 of file ReadFiles.cpp.

```
173 {
174     std::shorts::VV_string vv_strs;
175     for(auto const& line : lines)
176     {
177         auto v_str_tmp = parse_line(line, v_cols_start_length);
178         vv_strs.emplace_back(v_str_tmp);
179     }
180
181     // init n_rows and n_cols
182     n_rows = vv_strs.size();
183     n_cols = vv_strs[0].size();
184
185     headers.resize(n_cols);
186
187     // initializing headers
188     for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
189     {
190         if(is_first_col_header)
191         {
192             headers[i_col] = vv_strs[0][i_col];
193         }
194         else if(v_hdrs.size() > 0)
195         {
196             if(v_hdrs.size() == n_cols)
197             {
198                 headers[i_col] = v_hdrs[i_col];
199             }
200             else
201             {
202                 throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: number of provided
    headers does not match with the number of columns in the data"));
203             }
204         }
205         else
206         {
207             headers[i_col] = std::to_string(i_col + 1);
208         }
209     }
210
```

```
211     for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
212     {
213         std::shorts::V_string values;
214         for(unsigned long long i_row{is_first_col_header}; i_row < n_rows; ++i_row)
215         {
216             if(vv_strs[i_row].size() != n_cols)
217             {
218                 throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: inconsistent number of
    columns, check row {}", i_row + 1));
219             }
220
221             values.emplace_back(vv_strs[i_row][i_col]);
222
223             // check for the missig values
224             // missing values are empty string, NA, and NAN
225             if(vv_strs[i_row][i_col].empty() ||
226                vv_strs[i_row][i_col] == "NA" ||
227                vv_strs[i_row][i_col] == "NAN")
228             {
229                 mising_values.insert({i_row, i_col});
230             }
231         }
232
233         data[headers[i_col]] = values;
234     }
235 }
```

5.1.2.9 get_by_header()

```
std::shorts::V_string DF::DataFrame::get_by_header (
            std::string const & hdr )
```

Definition at line 360 of file ReadFiles.cpp.

```
361 {
362     return data[hdr];
363 }
```

5.1.2.10 get_headers()

```
std::shorts::V_string DF::DataFrame::get_headers ( ) const
```

Get the headers.

Returns

std::shorts::V_string headers

Definition at line 350 of file ReadFiles.cpp.

```
351 {
352     return headers;
353 }
```

5.1.2.11 get_n_cols()

```
int DF::DataFrame::get_n_cols ( ) const
```

Get the number cols of a given data.

Returns

int number of columns

Definition at line 11 of file ReadFiles.cpp.

```
12 {
13     return n_cols;
14 }
```

5.1.2.12 get_n_rows()

```
int DF::DataFrame::get_n_rows ( ) const
```

Get the number of rows of a given data.

Returns

int number of rows

Definition at line 16 of file ReadFiles.cpp.

```
17 {
18     return n_rows;
19 }
```

5.1.2.13 head()

```
void DF::DataFrame::head (
            unsigned long long n = 5 )
```

print n first rows off all columns

Parameters

n	

Definition at line 249 of file ReadFiles.cpp.

```
250 {
251     if(n > n_rows)
252     {
253         fmt::print(fg(fmt::color::yellow),"Warning: number of row requested ({}) to be printed is bigger
    than number of available rows in data ({})\n",
254         n, data[headers[0]].size());
255         if(n_rows > 5) n = 5;
256         else n = n_rows;
257     }
```

```
258
259     fmt::print("{:10}|", " ");
260     for(auto const& curr_hdr : headers)
261     {
262         fmt::print(fmt::emphasis::bold | fg(fmt::color::green), "{:^10}|", curr_hdr.substr(0,10));
263     }
264
265     std::string sub_separator(10, '-');
266     sub_separator += '+';
267
268     std::string separator;
269     separator.reserve(n_cols * sub_separator.size()+1);
270     for (size_t i = 0; i < n_cols+1; ++i) {
271         separator += sub_separator;
272     }
273
274     fmt::println("\n{}",std::string(separator));
275
276     // std::cout << '\n';
277
278     if(n <= 10)
279     {
280         for(unsigned long long i{0}; i < n; ++i)
281         {
282             fmt::print(fg(fmt::color::green),"|{:^9}|", i+1);
283             for(auto const& curr_hdr : headers)
284             {
285                 const auto& value = data[curr_hdr][i];
286                 if (value.empty() || value == "NA" || value == "NAN")
287                 {
288                     fmt::print(bg(fmt::color::red),"{:^10}", data[curr_hdr][i]);
289                     std::cout << "|";
290                 }
291                 else
292                 {
293                     fmt::print("{:^10}|", data[curr_hdr][i]);
294                 }
295
296             }
297
298             std::cout << "\n";
299         }
300     }
301     else
302     {
303         for(unsigned long long i{0}; i < 5; ++i)
304         {
305             fmt::print(fg(fmt::color::green),"|{:^9}|", i+1);
306             for(auto const& curr_hdr : headers)
307             {
308                 const auto& value = data[curr_hdr][i];
309                 if (value.empty() || value == "NA" || value == "NAN")
310                 {
311                     fmt::print(bg(fmt::color::red),"{:^10}", data[curr_hdr][i] /*data[curr_hdr][i]*/);
312                     std::cout << "|";
313                 }
314                 else
315                 {
316                     fmt::print("{:^10}|", data[curr_hdr][i] /*data[curr_hdr][i]*/);
317                 }
318
319             }
320
321             std::cout << "\n";
322         }
323
324         std::cout << "\n\t.\n\t.\n\t.\n\n";
325
326         for(unsigned long long i{n-5}; i < n; ++i)
327         {
328             fmt::print(fg(fmt::color::green),"|{:^9}|", i+1);
329             for(auto const& curr_hdr : headers)
330             {
331                 const auto& value = data[curr_hdr][i];
332                 if (value.empty() || value == "NA" || value == "NAN")
333                 {
334                     fmt::print(bg(fmt::color::red),"{:^10}", data[curr_hdr][i] /*data[curr_hdr][i]*/);
335                     std::cout << "|";
336                 }
337                 else
338                 {
339                     fmt::print("{:^10}|", data[curr_hdr][i] /*data[curr_hdr][i]*/);
340                 }
341
342             }
343
344             std::cout << "\n";
```

```
345         }
346     }
347     fmt::println("{}",std::string(separator));
348 }
```

5.1.2.14 insert_col()

```
void DF::DataFrame::insert_col (
            std::shorts::V_string const & values,
            std::string hdr )  [private]
```

Definition at line 391 of file ReadFiles.cpp.

```
392 {
393     auto [_it, inserted] = data.insert({hdr, values});
394 
395     int n = 1;
396     std::string original_hdr = hdr;
397 
398     while (!inserted)
399     {
400         hdr = fmt::format("{}_{}", original_hdr, n++);
401         std::tie(_it, inserted) = data.insert({hdr, values});
402     }
403     headers.push_back(hdr);
404     n_cols++;
405 }
```

5.1.2.15 operator[]()

```
std::shorts::V_string & DF::DataFrame::operator[] (
            std::string hdr )
```

subscript operator

Parameters

hdr	

Returns

std::shorts::V_string

Definition at line 444 of file ReadFiles.cpp.

```
445 {
446     return data[hdr];
447 }
```

5.1.2.16 parse_line() [1/2]

```
std::shorts::V_string DF::DataFrame::parse_line (
            std::string const & line,
            char delim )  [private]
```

Definition at line 64 of file ReadFiles.cpp.

```
65 {
66     std::shorts::V_string v_strs;
67 
68     std::istringstream iss(line);
69     std::string cell;
70 
71     while(std::getline(iss, cell, delim))
72     {
73         cell.erase(std::remove(cell.begin(), cell.end(), '\"'), cell.end());
74         cell.erase(std::remove(cell.begin(), cell.end(), ' '), cell.end());
75         v_strs.emplace_back(cell);
76     }
77 
78     return v_strs;
79 }
```

5.1.2.17 parse_line() [2/2]

```
std::shorts::V_string DF::DataFrame::parse_line (
            std::string const & line,
            std::shorts::V_pair_ints const & v_cols_start_ends )  [private]
```

Definition at line 81 of file ReadFiles.cpp.

```
82 {
83     std::shorts::V_string v_strs;
84 
85     for(auto const& [start, end] : v_cols_start_ends)
86     {
87         std::string cell;
88         try
89         {
90             cell = line.substr(start, end);
91         }
92         catch(...)
93         {
94             cell = "NA";
95         }
96 
97         // auto cell = line.substr(start, end);
98         cell.erase(std::remove(cell.begin(), cell.end(), '\"'), cell.end());
99         cell.erase(std::remove(cell.begin(), cell.end(), ' '), cell.end());
100         v_strs.push_back(cell);
101     }
102 
103     return v_strs;
104 }
```

5.1.2.18 read_files()

```
void DF::DataFrame::read_files (
            std::string_view path,
            char delim = ',',
            bool is_first_col_header = true,
            std::shorts::V_string v_hdrs = {} )
```

read files

Parameters

path	path to input file
delim	delimiter for parsing the input file
is_first_col_header	boolean

Definition at line 237 of file ReadFiles.cpp.

```
238 {
239     auto lines = read_lines(path);
240     fill_data(lines, delim, is_first_col_header, v_hdrs);
241 }
```

5.1.2.19 read_lines() [1/2]

```
std::shorts::V_string DF::DataFrame::read_lines (
            std::string const & text )  [private]
```

Definition at line 45 of file ReadFiles.cpp.

```
46 {
47     std::istringstream iss(text);
48 
49     std::string line;
50     std::shorts::V_string v_strs;
51 
52 
53     while(std::getline(iss, line))
54     {
55         if(line.size() == 0) continue;
56         line.erase(std::remove(line.begin(), line.end(), '\r'),line.end());
57         v_strs.emplace_back(line);
58     }
59 
60     return v_strs;
61 }
```

5.1.2.20 read_lines() [2/2]

```
std::shorts::V_string DF::DataFrame::read_lines (
            std::string_view path )  [private]
```

Definition at line 21 of file ReadFiles.cpp.

```
22 {
23     std::ifstream ifs(path.data());
24 
25     if(ifs.fail())
26     {
27         throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: unable to read file {}.\nPlease
       check your input.", path));
28     }
29 
30 
31     std::string line;
32     std::shorts::V_string v_strs;
33 
34 
35     while(std::getline(ifs, line))
36     {
37         if(line.size() == 0) continue;
38         line.erase(std::remove(line.begin(), line.end(), '\r'),line.end());
39         v_strs.emplace_back(line);
40     }
41 
42     return v_strs;
43 }
```

5.1.2.21 read_text()

```
void DF::DataFrame::read_text (
            std::string const & text,
            std::shorts::V_pair_ints const & v_cols_start_length,
            bool is_first_col_header = true,
            std::shorts::V_string v_hdrs = {} )
```

Parameters

path	std::string_view
v_cols_length	
is_first_col_header	

Definition at line 243 of file ReadFiles.cpp.

```
244 {
245     auto lines = read_lines(text);
246     fill_data(lines, v_cols_start_length, is_first_col_header, v_hdrs);
247 }
```

5.1.2.22 remove_duplications()

```
void DF::DataFrame::remove_duplications (
            std::string const & hdr )
```

get a given header and only keep the first occurance and remove the remaning rows

Parameters

hdr	header to check the duplication

5.1.2.23 save_as_csv()

```
void DF::DataFrame::save_as_csv (
            std::string_view path )
```

saving dataframe in comma separted format file

Parameters

path	

Definition at line 472 of file ReadFiles.cpp.

```
473 {
474     write(path /*,delimiter=','*/);
475 }
```

5.1.2.24 set_headers()

```
void DF::DataFrame::set_headers (
            std::shorts::V_string const & v_hdrs )
```

Set the headers with user provided vector of strings.

Parameters

v_hdrs	provided headers from users

Definition at line 355 of file ReadFiles.cpp.

```
356 {
357     headers = v_hdrs;
358 }
```

5.1.2.25 swap_cols_pos()

```
void DF::DataFrame::swap_cols_pos (
            std::string first_hdr,
            std::string second_hdr )
```

swap two columns with their header with each others

Parameters

first_hdr	std::string
second_hdr	std::string

Definition at line 380 of file ReadFiles.cpp.

```
381 {
382     std::string tmp_hdr;
383     std::shorts::V_string tmp_vals;
384     auto first_it = std::find(headers.begin(), headers.end(), first_hdr);
385     auto second_it = std::find(headers.begin(), headers.end(), second_hdr);
386     swap(headers[first_it - headers.begin()], headers[second_it - headers.begin()]);
387
388     std::swap(data[first_hdr], data[second_hdr]);
389 }
```

5.1.2.26 write()

```
void DF::DataFrame::write (
            std::string_view path,
            char delimiter = ',' )
```

write the dataframe in a file with given path and delimiter

Parameters

path	
delimiter	

Definition at line 449 of file ReadFiles.cpp.

```
450 {
451     fmt::ostream out = fmt::output_file(path.data());
452
```

```
453     for(auto const& curr_hdr : headers)
454     {
455         out.print("{}{}", curr_hdr.substr(0,10), delimiter);
456     }
457     out.print("\n");
458 
459     for(unsigned long long i{0}; i < n_rows; ++i)
460     {
461         for(auto const& curr_hdr : headers)
462         {
463             const auto& value = data[curr_hdr][i];
464             out.print("{}{}", value, delimiter);
465 
466         }
467 
468         out.print("\n");
469     }
470 }
```

5.1.3 Member Data Documentation

5.1.3.1 data

```
std::shorts::Data DF::DataFrame::data  [private]
```

Definition at line 215 of file ReadFiles.hpp.

5.1.3.2 headers

```
std::shorts::V_string DF::DataFrame::headers  [private]
```

Definition at line 216 of file ReadFiles.hpp.

5.1.3.3 mising_values

```
std::unordered_map<int, int> DF::DataFrame::mising_values
```

an unorderd maps for missing values

Definition at line 74 of file ReadFiles.hpp.

5.1.3.4 n_cols

```
unsigned long long DF::DataFrame::n_cols
```

number of columns

Definition at line 68 of file ReadFiles.hpp.

5.1.3.5 n_rows

```
unsigned long long DF::DataFrame::n_rows
```

number of rows

Definition at line 62 of file ReadFiles.hpp.

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

- include/ReadFiles.hpp
- src/ReadFiles.cpp

Chapter 6

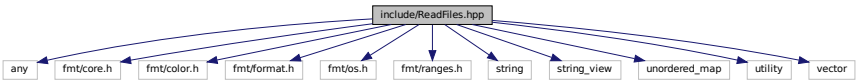
File Documentation

6.1 include/ReadFiles.hpp File Reference

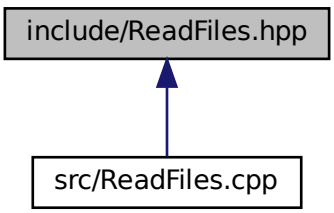
A class for reading files with different delimiters.

```
#include <any>
#include "fmt/core.h"
#include "fmt/color.h"
#include "fmt/format.h"
#include "fmt/os.h"
#include "fmt/ranges.h"
#include <string>
#include <string_view>
#include <unordered_map>
#include <utility>
#include <vector>
```

Include dependency graph for ReadFiles.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class DF::DataFrame

  DataFrame is class for parsing data in a given file with a give delimeter (default is comma ','). all data will be saved as string wich user can later covert to desired type.

Namespaces

- std
- std::shorts

  namespace for introducting shortnames
- DF

Typedefs

- using std::shorts::V_string = vector< string >
- using std::shorts::VV_string = vector< V_string >
- using std::shorts::V_any = vector< any >
- using std::shorts::VV_any = vector< vector< any > >
- using std::shorts::Data = unordered_map< string, V_string >
- using std::shorts::V_double = vector< double >
- using std::shorts::V_int = vector< int >
- using std::shorts::V_pair_ints = std::vector< std::pair< int, int > >

6.1.1 Detailed Description

A class for reading files with different delimiters.

Author

Naeim Moafinejad (snmoafinejad@iimcb.gov.pl, s.naeim.moafi.n@gmail.com)

Version

0.1

Date

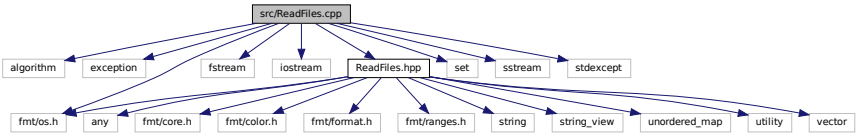
2024-10-29

Copyright

6.2 src/ReadFiles.cpp File Reference

```
#include <algorithm>
#include <exception>
#include <fmt/os.h>
#include <fstream>
#include <iostream>
#include "ReadFiles.hpp"
#include <set>
#include <sstream>
#include <stdexcept>
```

Include dependency graph for ReadFiles.cpp:



Functions

- int main ()

6.2.1 Function Documentation

6.2.1.1 main()

```
int main ( )
```

Definition at line 478 of file ReadFiles.cpp.

```
479 {
480     std::string software_name = R"(_____________                .__
481 \_   _____/____ _______|   |__ _____    ____    ____
482  |    __) \__  \\_  __ \   |  \\__  \  /    \  / ___\
483  |     \   / __ \|  | \/   Y  \/ __ \|   |  \/ /_/  >
484  \___  /  (____  /__|  |___|  (____  /___|  /\___  /
485      \/        \/           \/     \/     \//_____/ )";
486
487         std::cout << software_name << std::endl << std::endl;
488     DF::DataFrame df;
489
490
491     df.read_files("test.txt", '\t');
492     df.head();
493     std::cout << '\n';
494
495     auto df2 = df.copy_by_headers({"id", "age", "disease"});
496     df2.head();
497     std::cout << '\n';
498
499     df2.swap_cols_pos("age", "disease");
500     df2.head();
501     std::cout << '\n';
502
503     df.read_files("test.txt", '\t', false);
504     df.head();
505     std::cout << '\n';
```

```
506
507     df.read_files("test.txt", '\t', false, {"1.0000", "2.0000", "3.0000", "4.0000", "5.0000"});
508     df.head();
509     std::cout « '\n';
510
511     df.head(600);
512     std::cout « '\n';
513
514     DF::DataFrame df3;
515     std::vector<std::string> v_hdrs {"group_PDB", "id", "label_atom_id", "label_comp_id",
      "label_asym_id",
516                                      "label_seq_id", "Cartn_x", "Cartn_y", "Cartn_z", "occupancy",
517                                      "B_iso_or_equiv", "type_symbol", "charge"};
518
519     std::shorts::V_pair_ints fields_intervals = {{0,6}, {6,5}, {12,4}, {17,3}, {20,2}, {22,4}, {30,8},
      {38,8}, {46,8}, {54,6}, {60,6}, {76,2}, {78,2}};
520
521     // std::string text{"HETATM    1  PG  GTP A   1      24.181  32.064  27.670  0.10 24.73           P
      \nHETATM    2  O1G GTP A   1      24.342  33.433  27.064  0.10 37.56           O  \nHETATM    3  O2G
      GTP A   1      24.519  32.013  29.136  0.10 32.46           O  \n"};
522     std::string text = R"(HETATM    1  PG  GTP A   1      24.181  32.064  27.670  0.10 24.73           P

523 HETATM    2  O1G GTP A   1      24.342  33.433  27.064  0.10 37.56           O
524 HETATM    3  O2G GTP A   1      24.519  32.013  29.136  0.10 32.46           O
525 HETATM    4  O3G GTP A   1      25.048  31.062  26.907  1.00 47.91           O
526 HETATM    5  O3B GTP A   1      22.644  31.665  27.526  1.00 33.11           O )";
527
528     df3.read_text(text, fields_intervals, false, v_hdrs);
529     df3.head();
530
531
532     return EXIT_SUCCESS;
533 }
```

Index