
pandastable Documentation

Damien Farrell

Feb 06, 2023

CONTENTS

1	Introduction	3
2	Current Features	5
3	The DataExplore application	7
4	Links	9
5	Citation	11
6	Installation	13
6.1	For Dataexplore	13
6.2	pandastable library	13
6.3	Linux	14
6.4	Windows	14
6.5	Mac OSX	14
7	Using DataExplore	17
7.1	Purpose of the program	17
7.2	Table layout	17
7.3	Command Line	18
7.4	Import text files	18
7.5	Saving data	19
7.6	Getting table info	19
7.7	Cleaning data	19
7.8	String operations	20
7.9	Summarizing and grouping data	20
7.10	Merging two tables	20
7.11	Pivoting tables	21
7.12	Transpose tables	21
7.13	Filtering tables	21
7.14	Applying functions	22
7.15	Converting column names	22
7.16	Resampling columns	22
7.17	Plot options	22
7.18	Plotting grouped data	24
7.19	Plotting in a grid	24
7.20	Animated plots	24
7.21	Table Coloring	24
7.22	Setting preferences	25
7.23	Batch processing	26

7.24	Other examples	26
8	Code Examples	27
8.1	Basics	27
8.2	Sub-class the Table	28
8.3	Table methods	29
8.4	Accessing and modifying data directly	29
8.5	Set table attributes	30
8.6	Set Preferences	30
8.7	Table Coloring	31
8.8	Writing DataExplore Plugins	31
8.9	Freezing the app	32
9	pandatable	33
9.1	pandatable package	33
10	Indices and tables	35

Contents:

INTRODUCTION

The pandastable library provides a table widget for Tkinter with plotting and data manipulation functionality. It uses the pandas DataFrame class to store table data. Pandas is an open source Python library providing high-performance data structures and data analysis tools. Tkinter is the standard GUI toolkit for python. It is intended for the following uses:

- for python/tkinter GUI developers who want to include a table in their application that can store and process large amounts of data
- for non-programmers who are not familiar with Python or the pandas API and want to use the included DataExplore application to manipulate/view their data
- it may also be useful for data analysts and programmers who want to get an initial interactive look at their tabular data without coding

CURRENT FEATURES

- add, remove rows and columns
- spreadsheet-like drag, shift-click, ctrl-click selection
- edit individual cells
- sort by column, rename columns
- reorder columns dynamically by mouse drags
- set some basic formatting such as font, text size and column width
- save the DataFrame to supported pandas formats
- import/export of supported text files
- rendering of very large tables is only memory limited
- interactive plots with matplotlib, mostly using the pandas plot functions
- basic table manipulations like aggregate and pivot
- filter table using built in dataframe functionality
- graphical way to perform split-apply-combine operations

THE DATAEXPLORE APPLICATION

Installing the package creates a command *dataexplore* in your path. Just run this to open the program. This is a standalone application for data manipulation and plotting meant for education and basic data analysis. More details are in the ‘Using dataexplore’ section. Also see the home page for this application at <http://dmnfarrell.github.io/pandastable/>

CHAPTER FOUR

LINKS

<http://openresearchsoftware.metajnl.com/articles/10.5334/jors.94/>

<http://dmnfarrell.github.io/pandastable/>

<https://youtu.be/Ss0QIFywt74>

CITATION

If you use this software in your work please cite the following article:

Farrell, D 2016 DataExplore: An Application for General Data Analysis in Research and Education. Journal of Open Research Software, 4: e9, DOI: <http://dx.doi.org/10.5334/jors.94>

INSTALLATION

6.1 For Dataexplore

For **Windows users** there is an MSI installer for the DataExplore application. This is recommended for anyone using windows not using the library directly as a widget.

On linux snaps are highly recommended:

```
snap install dataexplore
```

6.2 pandastable library

On all operating systems installations of Python should include the pip tool. If not use your distributions package manager to install pip first. Then a simple call as follows should install all dependencies:

```
pip install pandastable
```

This might not work well in some cases because matplotlib has library dependencies that users might find confusing. Though it should work ok on recent versions of Ubuntu. Advice for each OS is given below.

Dependencies

- numpy
- pandas
- matplotlib
- numexpr

Optional dependencies

- statsmodels
- seaborn (requires scipy)

6.3 Linux

For the python linbrary using `easy_install` or `pip` should work well but for `matplotlib` might require more packages such as python headers for compiling the extension. You need the `tk8.6-dev` package to provide the `tkagg` backend.

Otherwise, to use the package manager in Ubuntu/Debian based distributions you can issue the command:

```
sudo apt install python-matplotlib
```

You should install `pandas` with `pip` as it will provide the most recent version. This will likely be done automatically anyway:

For python 3 installs

You need to use the command `pip3` instead if python 2 is also on your system, like in Ubuntu. When installing packages with `apt` you likely need to specify python 3. e.g. `python3-numpy` instead of `python-numpy`.

For python 2.7 ONLY

You will also need the `future` package. Run *pip install future* to install them. Python 2.6 has NOT been tested and probably won't work.

6.4 Windows

It is much easier to install `matplotlib` in windows using the binary installer rather than using `pip`. You can download this [here](<http://matplotlib.org/downloads.html>). Pick the appropriate file for your python version e.g. 'matplotlib-1.4.3.win32-py3.4.exe' for python 3.4.

`pandas` should install ok with the `pip` installer. In windows `pip.exe` is located under `C:Python34Scripts`. The `future` package is needed for python 2.7.

Note that the Python pydata stack can also be installed at once using `miniconda`, <http://conda.pydata.org/miniconda.html>. This includes a version of Python itself.

6.5 Mac OSX

There are multiple packaged installers for scientific Python, the best of which is probably `anaconda`. `Miniconda` is a smaller version if you don't want all the packages. To use it download and run the Mac OS X installer from <http://conda.pydata.org/miniconda.html>. The installer will automatically configure your system to use the `Anaconda Python`. You can then use `pip` to install the package.

If using `macports`:

```
sudo port install py34-pip
sudo pip-3.4 install matplotlib numpy pandas numexpr
```

Using the source distribution file

You can download the latest `tar.gz` file [here](<https://github.com/dmnfarrell/pandastable/releases/latest/>) and do the following:

```
tar -xzf pandastable.version.tar.gz
cd pandastable
sudo python3 setup.py install
```

Note that you still need to have installed the dependencies as above.

USING DATAEXPLORE

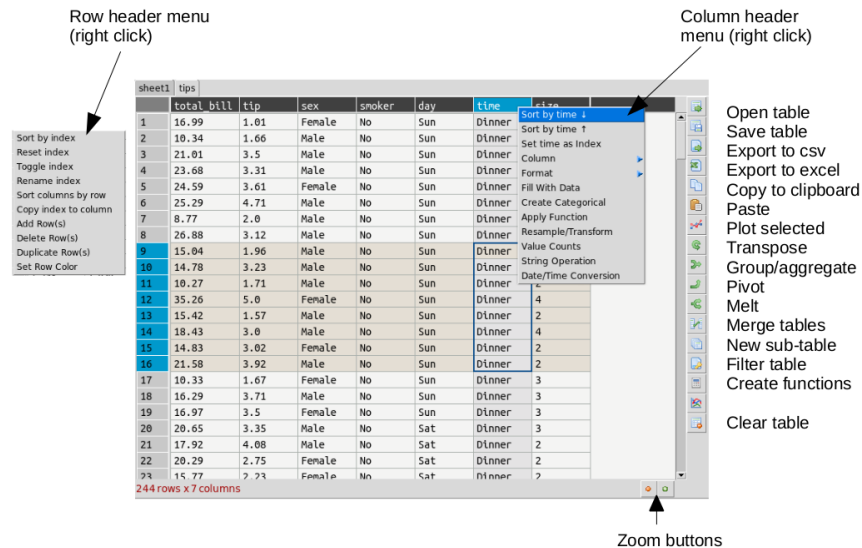
This page details some of the tasks available in dataexplore. For a general introduction also see the screencast at <https://youtu.be/Ss0QIFywt74>. Most of this functionality is available when you just use the table widget as well as the dataexplore application. Installing in windows or with a snap in linux should provide a menu item to launch the app. Otherwise use the command line, detailed below.

7.1 Purpose of the program

This program is for analyzing tabular data but is not meant to be a spreadsheet. Data is treated in a row/column centric fashion and a lot of the analysis is done in bulk on entire columns at once. So although you can edit cells it is not really meant for data entry. You can use a spreadsheet for that. Cell formulas are not possible for instance. You can however delete rows, columns and clear blocks of cells. New columns can be created through the use of functions. The primary goal is to let users explore their tables interactively without any prior programming knowledge and make interesting plots as they do this. One advantage is the ability to load and work with relatively large tables as compared to spreadsheets. So several million rows should not be a problem and is limited only by your computer memory.

7.2 Table layout

The table is laid out with headers for row and columns. Much functionality can be accessed from the tools menu but also by right clicking on the row and column headers. You can resize columns by dragging in the header. Rows cannot be resized independently (zoom in to enlarge). Unlike spreadsheets column and row headers can use indexes. You can set any column as an index. This has extra functionality when it comes to plotting.



7.3 Command Line

Launching dataexplore from the command line allows you to provide several options using unix type '-' switches.

Show help:

```
dataexplore -h
```

Open a project file:

```
dataexplore -p <project file>
```

Open a dataframe stored as a messagepack file:

```
dataexplore -f <msgpack file>
```

Open a csv file and try to import it:

```
dataexplore -i <csv file>
```

Launch a basic test table with no plot frame

```
dataexplore -t
```

7.4 Import text files

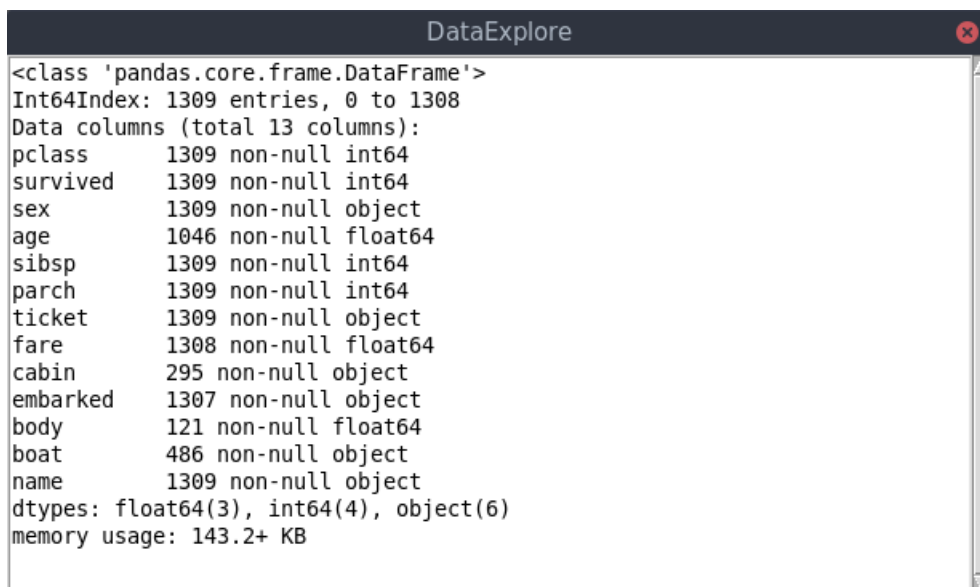
Import of csv and general plain text formats is done from the file menu, toolbar or by right-clicking anywhere in the table and using the context menu. The dialog has most basic options such as delimiter, header selection, comment symbol, rows to skip etc. When you change the import option you can update the preview to see if the new table will look correct. You then press import. Note that it is generally a good idea to remove empty lines and bad data if you can before importing.

7.5 Saving data

Dataexplore projects (multiple groups of sheets with the plot view for each) are saved in **messagepack** format and have the .dexpl file extension. Tables can also be saved on their own as messagepack or pickle files and then opened directly in Python. Using the messagepack format is more efficient than csv as it takes up less space and loads faster. Though quite reliable and efficient, it is not recommended that you use these formats for long term backup, *always keep a copy of your raw data* if it is important. Exporting to csv is also possible and saving individual tables to excel.

7.6 Getting table info

The status bar at the bottom left shows the size of the table in rows and columns at all times. For a more detailed summary use Tools->Table info. This brings up a window showing the type of each column and memory usage. 'object' columns are those with text/mixed data and float and int must be numbers only.



```

DataExplore
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 1308
Data columns (total 13 columns):
pclass      1309 non-null int64
survived     1309 non-null int64
sex          1309 non-null object
age          1046 non-null float64
sibsp        1309 non-null int64
parch        1309 non-null int64
ticket       1309 non-null object
fare         1308 non-null float64
cabin        295 non-null object
embarked     1307 non-null object
body         121 non-null float64
boat         486 non-null object
name         1309 non-null object
dtypes: float64(3), int64(4), object(6)
memory usage: 143.2+ KB

```

7.7 Cleaning data

Pandas supports a variety of options for data 'cleaning' or dealing with missing data. The most basic are available from DataExplore from the main menu.

- Drop rows/columns with missing (empty) data
- Fill missing data with a symbol
- Forward or backfill with neighbouring row values
- Interpolate missing data (filling in the points between)
- Drop duplicates

7.8 String operations

Accessed by right clicking on the column header menu. String operations can be carried out on any column as long as they are object data types and not pure numbers.

The following string methods are supported:

- split, with separator symbol - will create multiple new columns
- strip, remove whitespace
- lower/upper case conversion
- title, convert to TitleCase
- swap case
- get length of string
- concat, concatenate strings in first two cols with given separator
- slice, slice string by start/end indexes
- replace

7.9 Summarizing and grouping data

For overall table statistics you can use the tools->describe table command. For individual columns you can get value counts by right clicking on the header.

The primary way to summarize data is to use the aggregate dialog. It is accessed on the right toolbar. Tables can be grouped and aggregated on multiple columns to create new summary tables. The results will be placed in the sub table below the main one and can then be copied to new sheets. Normally you would group by category columns (rather than a continuous variable like decimal numbers). The dialog has a list of columns to group by and another list box for column(s) to aggregate these groups using one or more functions. See the animated example (click to enlarge):

It is often easiest to test the selections out until you get the required result.

7.10 Merging two tables

Merging tables is done in dataexplore by first putting your second table in the sub-table below. You can do that by pasting it from another sheet or making an empty sub-table and importing. Once this is done you open the merge dialog in the toolbar. You select which columns in each table to merge on (at least one columns should be shared between each). The apply and the result is opened in the dialog to preview. You can copy this to a new sheet.

7.11 Pivoting tables

Pivot tables is an operation some people might be familiar with from excel. A pivot might best be described as way of summarizing data by ‘unstacking’ the grouped data into new columns. It is a more specialized version of the aggregation method above. A comprehensive explanation is given here: <https://www.dataquest.io/blog/pandas-pivot-table/>

7.12 Transpose tables

A transpose is rotating the table on its axes so the rows become columns and vice versa. This can be useful for plotting purposes when you want to treat the row data as series. This is illustrated in the animation below where the same table is plotted first with the years as series and then with ‘col1’ and ‘col2’ as series and years as data points. Your row index will become the new columns when you transpose, so you should make sure the **correct index is set** beforehand. If you make a mistake you can undo or transpose again to reverse. Note: transposing extremely large tables might be slow.

7.13 Filtering tables

Filtering tables is done using either a string query and/or one or more pre-defined filters defined with widgets.

7.13.1 Query with widgets

Pressing the filtering button will bring up the dialog below the table. Manual predefined filters can be added by pressing the + button. These are used alone or in conjunction with the string query as shown below. The filters are joined together using the first menu item using either ‘AND’, ‘OR’ or ‘NOT’ boolean logic. When filtered results are found the found rows are highlighted. You can also limit the table to show the filtered set which can be treated as usual (i.e. plots made etc). Closing the query box restores the full table. If you want to keep the filtered table you can copy and paste in another sheet.

7.13.2 String query

String based query are made up fairly intuitive expressions. The one caveat is that column names cannot contain spaces to be used in an expression. It is best in these cases to convert column names (i.e. replace spaces with an underscore ‘_’). You may also use Python/pandas style expressions to perform filters, useful with string based queries.

Examples:

```
x>4 and y<3 #filter by values of columns x and y
x.str.contains("abc") #find only values of column x containing substring #abc
x.str.len()>3 #find only rows where length of strings in x is greater than 3
```

7.14 Applying functions

Unlike a spreadsheet there are no cell based formulas. Rather functions are applied to columns over all rows, creating a new column. New columns can be created in several ways through computations on other columns. The column header menu provides some of these like resample/transform a column or the apply function dialog. Another more general way to add functions is to use the calculation button on the toolbar. This brings up a dialog below the table where you can type function as text expressions.

The same as for filtering, a string is entered like a formula and if it can be parsed a new column is created. For example entering 'x = a + b' will create a new column x that is the sum of a and b.

Examples:

```
x = a+b  #sum a and b
x = a*a  #a squared
x = sin(a)
x = sqrt(a+b)/log(c)
```

Supported functions in expressions: sin, cos, tan, arcsin, arccos, arctan, sinh, cosh, tanh, log, log10, exp

7.15 Converting column names

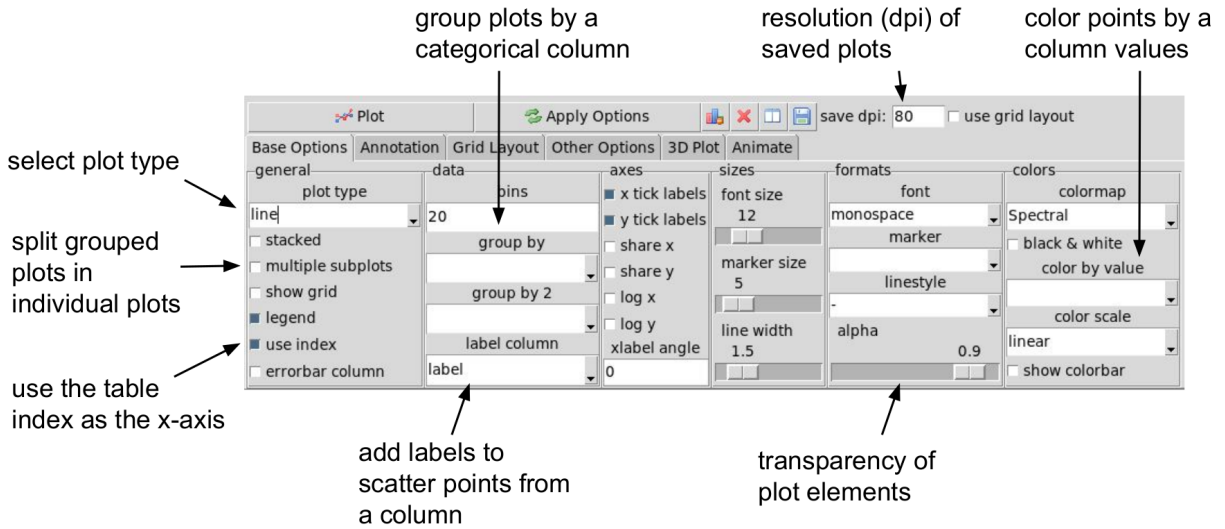
It may sometimes be necessary to re-format column names, for example to remove unwanted characters. If you have dozens or more columns this would be time consuming, so there is a function in dataexplore to do this in one step. Accessed from Tools->Convert column names, this dialog allows you to replace characters e.g. replace spaces with '_' symbol. You can also convert cases.

7.16 Resampling columns

Resampling is a way to average data over specific windows or periods. It is a possible way to smooth out noisy data for example or get an average trend. You can resample columns from the column header menu. In the example below this is used to smooth out the sawtooth shaped C02 data. The larger the window the more averaging will take place.

7.17 Plot options

The plot frame has an options dialog underneath with multiple tabs grouped by functionality. Most default formatting options such as the type of plot, whether to show a legend etc. are in the first tab. The dialogs may look a bit cluttered for some users but the idea is to have all available options quickly accessible rather than hidden in menus. If you use the program regularly you will be familiar with where things are. Some of the less obvious options are explained in the image below.



The following plot types are currently supported:

- line
- scatter
- bar
- barh
- pie
- histogram
- box plot
- dot plot
- heatmap
- area
- hexbin
- contour
- scatter matrix
- venn

Other tabs contain options for grid layouts, text annotation such as titles and text boxes, and access to the plot animation settings.

7.18 Plotting grouped data

Rather than grouping the table directly it is also possible to plot data grouped. This requires you select the appropriate columns including the one to be grouped by and select the grouping column in the ‘groupby’ menu in the plot options. Plots can be grouped by 2 columns at once.

7.19 Plotting in a grid

The gif animation below shows how to use the grid layout tool to generate subplots by clicking and dragging in the grid to select the area for your next plot. Note that subplots will be overwritten if you select the same cell as one currently occupied but if you drag over this cell the region will be plotted over. The tool assumes the user will know how to avoid overlaps. So it’s best to have a good idea of how to layout the plots beforehand, or just use trial and error. You can remove subplots from the drop down menu, listed according to their positions.

Grid layout includes other modes ‘split data and ‘multiview’. Split data lets you pick a grid size and splits up the rows into chunks, plotting each separately. The multiview mode allows you to auto-generate different kinds of plot in the grid for the same data every time you plot. This could be useful for quickly previewing regions of data repeatedly without having to set the plot type each time. This will overwrite whatever plot you currently have displayed. The feature is also illustrated in the gif above.

7.20 Animated plots

Plots can be animated and save as video files using the plot animation options tab. This would mostly be useful for time series based line plots but any kinds of plots can be animated. Formatting can be changed or column selections altered as the plot is updating, leading to some odd plot displays.

see <http://dmnfarrell.github.io/dataexplore/2018/05/15/animation>

7.21 Table Coloring

Column colors can be set by right clicking in the column header and choosing ‘set color’. A color picker will open. The formatting is saved with the project file. You can clear the formatting from the table popup menu.

You can set row and cell colors in several ways. Firstly, if right clicking on the row header or inside the table the ‘set color’ option lets you color the selected rows/columns with a single color. You can also set colors for the entire table/column according to the cell values. This is done by choosing ‘color by value’ from the column header and will allow you to select a color map. String values will be mapped to categorical integers and then to colors. See below:

	class	sepal_leng	sepal_widt	petal_lengt	petal_width
17	virginica	6.9	3.1	5.1	2.3
18	virginica	6.8	3.2	5.9	2.3
19	versicolor	6.8	2.8	4.8	1.4
20	virginica	6.8	3.0	5.5	2.1
21	versicolor	6.7	3.1	4.7	1.5
22	versicolor	6.7	3.0	5.0	1.7
23	virginica	6.7	3.1	5.6	2.4
24	virginica	6.7	3.3	5.7	2.5
25	virginica	6.7	2.5	5.8	1.8
26	virginica	6.7	3.0	5.2	2.3
27	virginica	6.7	3.3	5.7	2.1
28	versicolor	6.7	3.1	4.4	1.4
29	versicolor	6.6	2.9	4.6	1.3
30	versicolor	6.6	3.0	4.4	1.4
31	virginica	6.5	3.0	5.5	1.8
32	versicolor	6.5	2.8	4.6	1.5
33	virginica	6.5	3.2	5.1	2.0
34	virginica	6.5	3.0	5.8	2.2
35	virginica	6.5	3.0	5.2	2.0
36	virginica	6.4	2.8	5.6	2.2
37	virginica	6.4	3.1	5.5	1.8
38	virginica	6.4	3.2	5.3	2.3
39	virginica	6.4	2.7	5.3	1.9
40	versicolor	6.4	2.9	4.3	1.3

For very large tables, adding colors for all cells will increase the file size of saved projects.

7.22 Setting preferences

Preferences for table formatting can be set from the edit->preferences menu item. This uses a text configuration file stored in ~/.dataexplore/default.conf. The preferences dialog is used to apply the settings to the current table and/or save them to this file. This file can be edited manually in a text editor if you wish. Any new tables will use these settings. The file looks like this:

```
[base]
align = w
cellwidth = 80
floatprecision = 2
font = Arial
fontsize = 12
linewidth = 1
rowheight = 22

[colors]
cellbackgr = #F4F4F3
grid_color = #ABB1AD
rowselectedcolor = #E4DED4
textcolor = black
```

7.23 Batch processing

A plugin provides the ability to batch import and/or plot multiple files at once. This is generally designed for many similarly formatted files that you wish to clean or plot in bulk without loading each individually. You can also use this to join many files into one table. Access this tool from Plugins->Batch Process.

7.24 Other examples

Other guides are available as blog posts:

- <http://dmnfarrell.github.io/dataexplore/titanic-example>
- <http://dmnfarrell.github.io/dataexplore/grouped-plots>
- <http://dmnfarrell.github.io/dataexplore/sea-ice-example>

CODE EXAMPLES

This section is for python programmers you want to use the table widget in their own programs.

8.1 Basics

Create a parent frame and then add the table to it:

```
from tkinter import *
from pandastable import Table
#assuming parent is the frame in which you want to place the table
pt = Table(parent)
pt.show()
```

Update the table:

```
#alter the DataFrame in some way, then update
pt.redraw()
```

Import a csv file:

```
pt.importCSV('test.csv')
```

A class for launching a basic table in a frame:

```
from tkinter import *
from pandastable import Table, TableModel, config

class TestApp(Frame):
    """Basic test frame for the table"""
    def __init__(self, parent=None):
        self.parent = parent
        Frame.__init__(self)
        self.main = self.master
        self.main.geometry('600x400+200+100')
        self.main.title('Table app')
        f = Frame(self.main)
        f.pack(fill=BOTH, expand=1)
        df = TableModel.getSampleData()
        self.table = pt = Table(f, dataframe=df,
                                showtoolbar=True, showstatusbar=True)
```

(continues on next page)

(continued from previous page)

```

        pt.show()
        #set some options
        options = {'colheadercolor':'green','floatprecision': 5}
        config.apply_options(options, pt)

    pt.show()
    return

app = TestApp()
#launch the app
app.mainloop()

```

8.2 Sub-class the Table

Override the right click popup menu:

```

class MyTable(Table):
    """Custom table class inherits from Table. You can then override required methods"""
    def __init__(self, parent=None, **kwargs):
        Table.__init__(self, parent, **kwargs)
        return

        def handle_left_click(self, event):
            """Example - override left click"""

            Table.handle_left_click(self, event)
#do custom code here
        return

    def popupMenu(self, event, rows=None, cols=None, outside=None):
        """Custom right click menu"""

        popupmenu = Menu(self, tearoff = 0)
        def popupFocusOut(event):
            popupmenu.unpost()

            # add commands here
# self.app is a reference to the parent app
        popupmenu.add_command(label='do stuff', command=self.app.stuff)
        popupmenu.bind("<FocusOut>", popupFocusOut)
        popupmenu.focus_set()
        popupmenu.post(event.x_root, event.y_root)
        return popupmenu

```


8.3 Table methods

You can use the Table class methods to directly access data and perform many more functions. Check the API for all the methods. Some examples are given here:

```
#add 10 empty columns
table.autoAddColumns(10)
#resize the columns to fit the data better
table.autoResizeColumns()
#clear the current formatting
table.clearFormatting()
#reduce column widths proportionally
table.contractColumns()
#get selected column
table.getSelectedColumn()
#sort by column index 0
table.sortTable(0)
#enlarge all table elements
table.zoomIn()
#set row colors
table.setRowColors(rows=range(2,100,2), clr='lightblue', cols='all')
#delete selected rows
table.setSelectedRows([[4,6,8,10]])
#delete current row
table.deleteRow()
#set current row
table.setSelectedRow(10)
#insert below current row
table.insertRow()
```

8.4 Accessing and modifying data directly

The tables use a pandas DataFrame object for storing the underlying data. If you are not familiar with pandas you should learn the basics if you need to access or manipulate the table data. See <http://pandas.pydata.org/pandas-docs/stable/10min.html>

Each table has an object called model with has the dataframe inside it. The dataframe is referred to as df. So to access the data on a table you can use:

```
df = table.model.df
```

Examples of simple dataframe operations. Remember when you update the dataframe you will need to call table.redraw() to see the changes reflected:

```
df.drop(0) #delete column with this index
df.T #transpose the DataFrame
df.drop(columns=['x'])
```

8.5 Set table attributes

You can set table attributes directly such as these examples:

```
table.textcolor = 'blue'
table.cellbackgr = 'white'
table.boxoutlinecolor = 'black'
#set header colors
self.table.rowheader.bgcolor = 'orange'
self.table.colheader.bgcolor = 'lightgreen'
self.table.colheader.textcolor = 'black'
#make editable or not
table.editable = False
```

8.6 Set Preferences

Preferences are normally loaded from a configuration file that can be edited manually or via the menu. You can also programmatically set these preferences using the config module:

```
#load from a config file if you need to (done by default when tables are created)
options = config.load_options()
#options is a dict that you can set yourself
options = {'floatprecision': 2}
config.apply_options(options, table)
```

You can set the following configuration values:

```
{'align': 'w',
 'cellbackgr': '#F4F4F3',
 'cellwidth': 80,
 'floatprecision': 2,
 'thousandseparator': '',
 'font': 'Arial',
 'fontsize': 12,
 'fontstyle': '',
 'grid_color': '#ABB1AD',
 'linewidth': 1,
 'rowheight': 22,
 'rowselectedcolor': '#E4DED4',
 'textcolor': 'black'}
```

8.7 Table Coloring

You can set column colors by setting the key in the columncolors dict to a valid hex color code. Then just redraw:

```
table.columncolors['mycol'] = '#dcf1fc' #color a specific column
table.redraw()
```

You can set row and cell colors in several ways. `table.rowcolors` is a pandas dataframe that mirrors the current table and stores a color for each cell. It only adds columns as needed. You can update this manually but it is easiest to use the following methods:

```
table.setRowColors(rows, color) #using row numbers
table.setColorByMask(column, mask, color) #using a pre-defined mask
table.redraw()
```

To color by column values:

```
table.multiplecollist = [cols] #set the selected columns
table.setColorbyValue()
table.redraw()
```

To clear formatting:

```
table.clearFormatting()
table.redraw()
```

Note: You should generally use a simple integer index for the table when using colors as there may be inconsistencies otherwise.

8.8 Writing DataExplore Plugins

Plugins are for adding custom functionality that is not present in the main application. They are implemented by subclassing the Plugin class in the plugin module. This is a python script that can generally contain any code you wish. Usually the idea will be to implement a dialog that the user interacts with. But this could also be a single method that runs on the current table or all sheets at once.

8.8.1 Implementing a plugin

Plugins should inherit from the Plugin class. Though this is not strictly necessary for the plugin to function.

```
from pandastable.plugin import Plugin
```

You can simply copy the example plugin to get started. All plugins need to have a `main()` method which is called by the application to launch them. By default this method contains the `_doFrame()` method which constructs a main frame as part of the current table frame. Usually you override `main()` and call `_doFrame` then add your own custom code with your widgets.

`_doFrame` method has the following lines which are always needed unless it is a non GUI plugin:

```
self.table = self.parent.getCurrentTable() #get the current table
#add the plugin frame to the table parent
self.mainwin = Frame(self.table.parentframe)
```

(continues on next page)

(continued from previous page)

```
#pluginrow is 6 to make the frame appear below other widgets
self.mainwin.grid(row=pluginrow,column=0,columnspan=2,sticky='news')
```

You can also override the `quit()` and `about()` methods.

8.8.2 Non-table based plugins

Plugins that don't rely on using the table directly do not need to use the above method and can have essentially anything in them as long as there is a `main()` method present. The Batch File Rename plugin is an example. This is a standalone utility launched in a separate toplevel window.

see <https://github.com/dmnfarrell/pandastable/blob/master/pandastable/plugins/rename.py>

8.9 Freezing the app

Dataexplore is available as an exe with msi installer for Windows. This was created using the `cx_freeze` package. For anyone wishing to freeze their tkinter app some details are given here. This is a rather hit and miss process as it seems to depend on your installed version of Python. Even when the msi/exe builds you need to check for runtime issues on another copy of windows to make sure it's working. Steps:

- Use a recent version of python (≥ 3.6 recommended) installed as normal and then using pip to install the dependencies that you normally need to run the app.
- The freeze script is found in the main pandastable folder, `freeze.py`. You can adopt it for your own app.
- Run the script using `python freeze.py bdist_msi`
- The resulting msi is placed in the dist folder. This is a 32 bit binary but should run fine on windows 10.

You can probably use Anaconda to do the same thing but we have not tested this.

PANDASTABLE

9.1 pandastable package

9.1.1 Submodules

9.1.2 pandastable.annotation module

9.1.3 pandastable.app module

9.1.4 pandastable.core module

9.1.5 pandastable.data module

9.1.6 pandastable.dialogs module

9.1.7 pandastable.handlers module

9.1.8 pandastable.headers module

9.1.9 pandastable.images module

9.1.10 pandastable.plotting module

9.1.11 pandastable.plugin module

9.1.12 pandastable.config module

9.1.13 pandastable.stats module

9.1.14 pandastable.tests module

9.1.15 pandastable.util module

9.1.16 Module contents

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`