tablexplore Documentation

Damien Farrell

May 22, 2022

CONTENTS

Contents:

CHAPTER ONE

INTRODUCTION

Tablexplore is an application for data analysis and plotting built in Python using the PySide2/Qt toolkit. It uses the pandas DataFrame class to store the table data. Pandas is an open source Python library providing high-performance data structures and data analysis tools.

This application is intended primarily for educational/scientific use and allows quick visualization of data with convenient plotting. 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. The focus is on data manipulation rather than data entry. Though basic cell editing and row/column changes are supported.

CHAPTER TWO

INSTALLATION

For all operating systems with Python and pip installed:

```
pip install -e git+https://github.com/dmnfarrell/tablexplore.git#egg=tablexplore
```

2.1 Linux

There is also a [snap](https://snapcraft.io/tablexplore) available, which can be installed using:

```
snap install tablexplore
```

2.2 Windows

A Windows standalone binary can be downloaded [here](https://dmnfarrell.github.io/tablexplore/).

CHAPTER

THREE

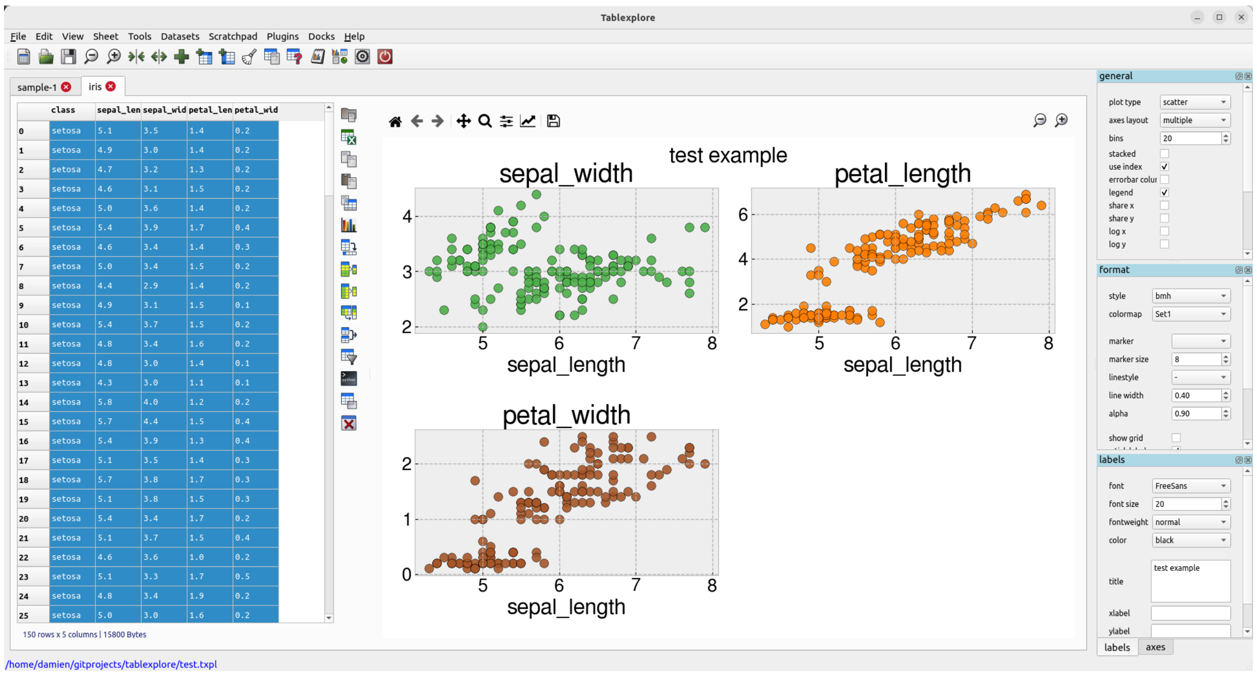
CURRENT FEATURES

- save and load projects
- import csv/hdf/from urls
- delete/add columns
- groupby-aggregate/pivot/transpose/melt operations
- merge tables
- show sub-tables
- plotting mostly works
- apply column functions, resample, transform, string methods and date/time conversion
- python interpreter

CHAPTER

FOUR

SCREENSHOTS



	class	sepal_len	sepal_wid	petal_len	petal_wid
0	setosa	5.1	3.5	1.4	0.2
1	setosa	4.9	3.0	1.4	0.2
2	setosa	4.7	3.2	1.3	0.2
3	setosa	4.6	3.1	1.5	0.2
4	setosa	5.0	3.6	1.4	0.2
5	setosa	5.4	3.9	1.7	0.4
6	setosa	4.6	3.4	1.4	0.3
7	setosa	5.0	3.4	1.5	0.2
8	setosa	4.4	2.9	1.4	0.2
9	setosa	4.9	3.1	1.5	0.1
10	setosa	5.4	3.7	1.5	0.2
11	setosa	4.8	3.4	1.6	0.2
12	setosa	4.8	3.0	1.4	0.1
13	setosa	4.3	3.0	1.1	0.1
14	setosa	5.8	4.0	1.2	0.2
15	setosa	5.7	4.4	1.5	0.4
16	setosa	5.4	3.9	1.3	0.4
17	setosa	5.1	3.5	1.4	0.3
18	setosa	5.7	3.8	1.7	0.3
19	setosa	5.1	3.8	1.5	0.3
20	setosa	5.4	3.4	1.7	0.2
21	setosa	5.1	3.7	1.5	0.4
22	setosa	4.6	3.6	1.0	0.2
23	setosa	5.1	3.3	1.7	0.5
24	setosa	4.8	3.4	1.9	0.2
25	setosa	5.0	3.0	1.6	0.2

150 rows x 5 columns | 15800 Bytes

/home/damien/gitprojects/tablexplore/test.txpl

general

plot type	scatter
axes layout	multiple
bins	20
stacked	☐
use index	✓
errorbar colur	☐
legend	✓
share x	☐
share y	☐
log x	☐
log y	☐

format

style	bmh
colormap	Set1
marker	
marker size	8
linestyle	-
line width	0.40
alpha	0.90
show grid	☐

labels

font	FreeSans
font size	20
fontweight	normal
color	black
title	test example
xlabel	
ylabel	

CHAPTER

FIVE

VIDEOS

- Introduction
- Summarizing Data
- Plotting
- Table Filtering
- Joining tables

CHAPTER

SIX

USING THE PROGRAM

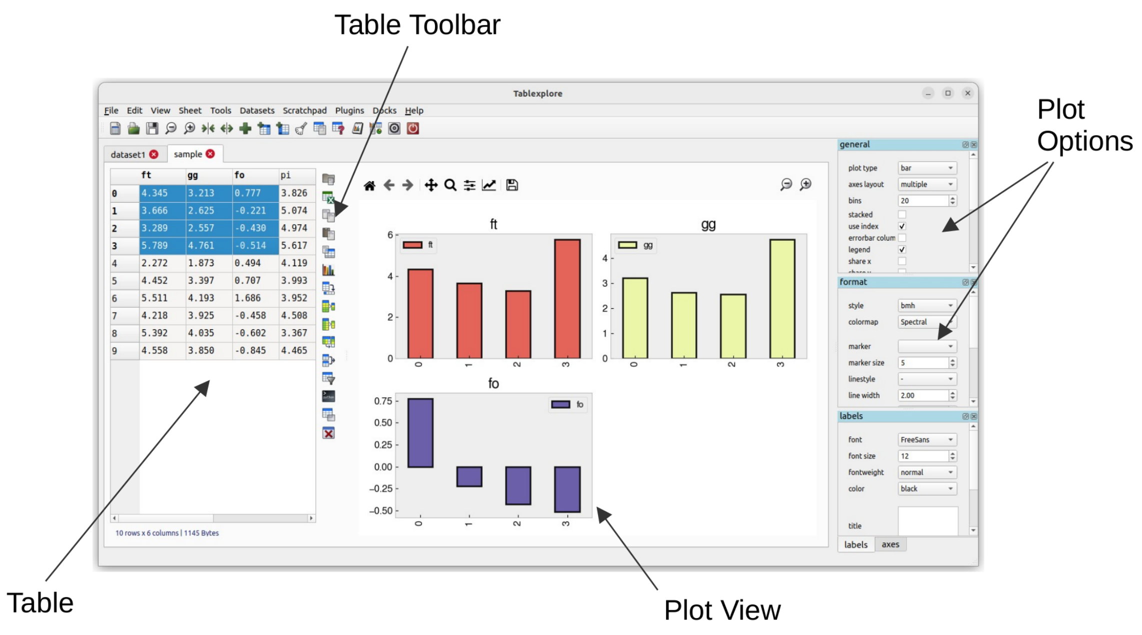
This page details some of the tasks available in tablexplore. For a general introduction also see the screencast at https://youtu.be/Ss0QIFywt74. Most of this functionality is available when you just use the table widget as well as the tablexplore application. Installing in windows or with a snap in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

6.1 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 rows/columns at once. So although you can edit cells it is not especially meant for data entry. You can use a spreadsheet for that and then import the data. 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 basic 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.

6.2 Interface 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 (arbitrary labels). You can set any column as an index. This has extra functionality when it comes to plotting. Duplicate column name indexes are allowed though should be avoided if you want predictable behaviour. Every table has a vertical toolbar on the right and a plot view associated with it. You can make the plot hidden if needed.



	ft	gg	fo	pi
0	4.345	3.213	0.777	3.826
1	3.666	2.625	-0.221	5.074
2	3.289	2.557	-0.430	4.974
3	5.789	4.761	-0.514	5.617
4	2.272	1.873	0.494	4.119
5	4.452	3.397	0.707	3.993
6	5.511	4.193	1.686	3.952
7	4.218	3.925	-0.458	4.508
8	5.392	4.035	-0.602	3.367
9	4.558	3.850	-0.845	4.465

10 rows x 6 columns | 1145 Bytes

general

plot type	bar
axes layout	multiple
bins	20
stacked	☐
use index	✓
errorbar colum	☐
legend	✓
share x	☐

format

style	bmh
colormap	Spectral
marker	
marker size	5
linestyle	-
line width	2.00

labels

font	FreeSans
font size	12
fontweight	normal
color	black
title	

6.3 Command Line

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

Show help:

```
tablexplore -h
```

Open a project file:

```
tablexplore -p <project file>
```

Open a csv file and try to import it:

```
tablexplore -i <csv file>
```

Open an excel file and try to import it:

```
tablexplore -x <excel file>
```

6.4 Import text files

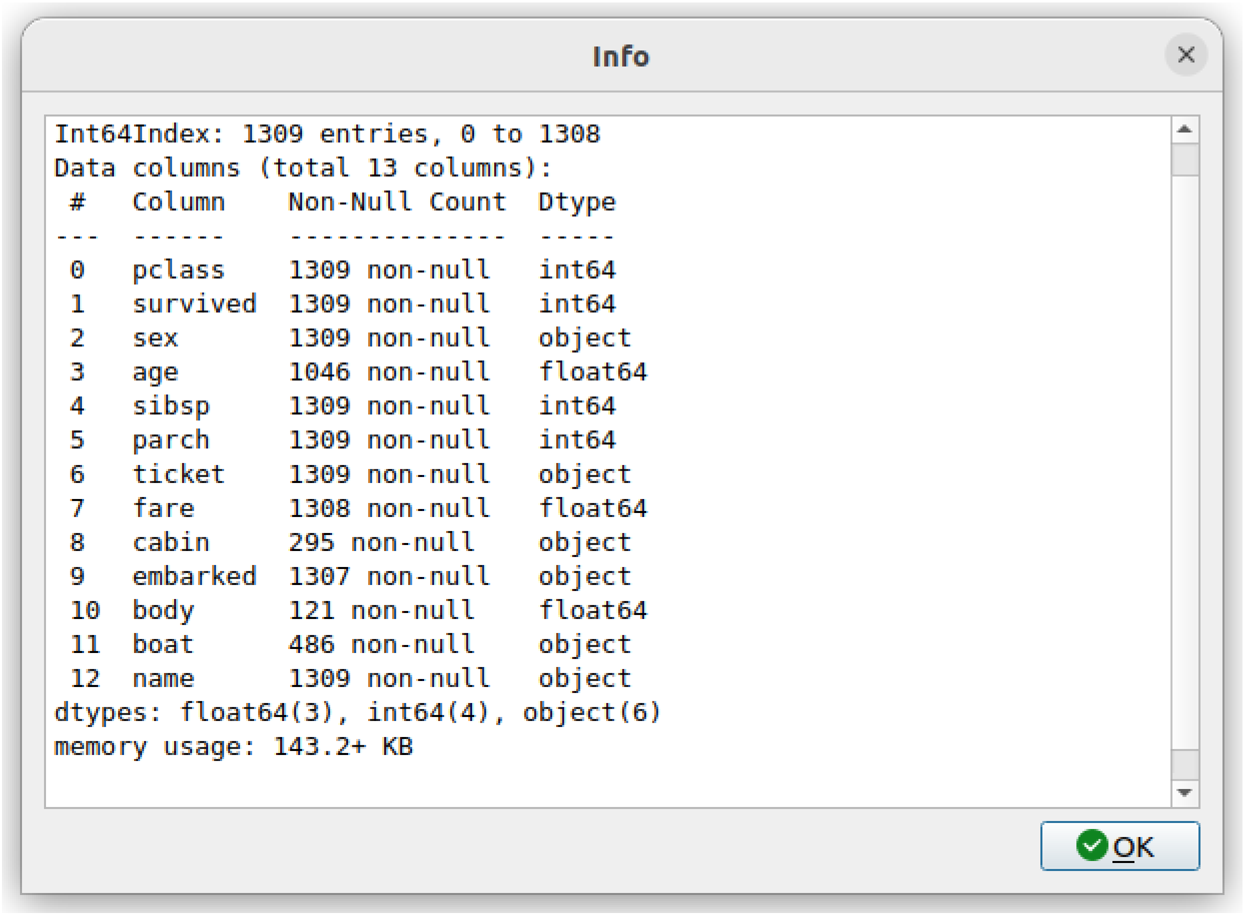
Import of csv and general plain text formats is done from the file menu, toolbar or by right-clicking anywhere in the table and using the context menu. The dialog has most basic options such as delimiter, header selection, comment symbol, rows to skip etc. When you change the import option you can update 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.

6.5 Saving your work

tablexplore projects (multiple groups of sheets with the plot and settings for each) are saved in **pickle** format and have the .txpl file extension. The program will remember table selections also. Note: it is not recommended that you use this project format for long term backup as it is subject to change with new versions. You should *always keep a copy your raw data* if it is important. Exporting to csv is also possible and saving individual tables to excel files.

6.6 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.



```
Int64Index: 1309 entries, 0 to 1308
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   pclass    1309 non-null   int64
 1   survived  1309 non-null   int64
 2   sex       1309 non-null   object
 3   age       1046 non-null   float64
 4   sibsp     1309 non-null   int64
 5   parch     1309 non-null   int64
 6   ticket    1309 non-null   object
 7   fare      1308 non-null   float64
 8   cabin     295 non-null    object
 9   embarked  1307 non-null   object
 10  body      121 non-null    float64
 11  boat      486 non-null    object
 12  name      1309 non-null   object
dtypes: float64(3), int64(4), object(6)
memory usage: 143.2+ KB
```

6.7 Cleaning data

Pandas supports a variety of options for data 'cleaning' or dealing with missing data. The most basic are available in tablexplore 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

6.8 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

6.9 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 then 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.

6.10 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/ The example below shows the titanic data being pivoted to show average ages per sex by pclass.

6.11 Merging two tables

Merging tables is done in tablexplore 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.

6.12 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. 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.

6.13 Filtering tables

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

6.13.1 Filter 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.

6.13.2 String filter

String based queries are made up fairly intuitive expressions in Python syntax. 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 of string filters:

```
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
```

Example of usage:

6.14 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.

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

6.15 Converting column names

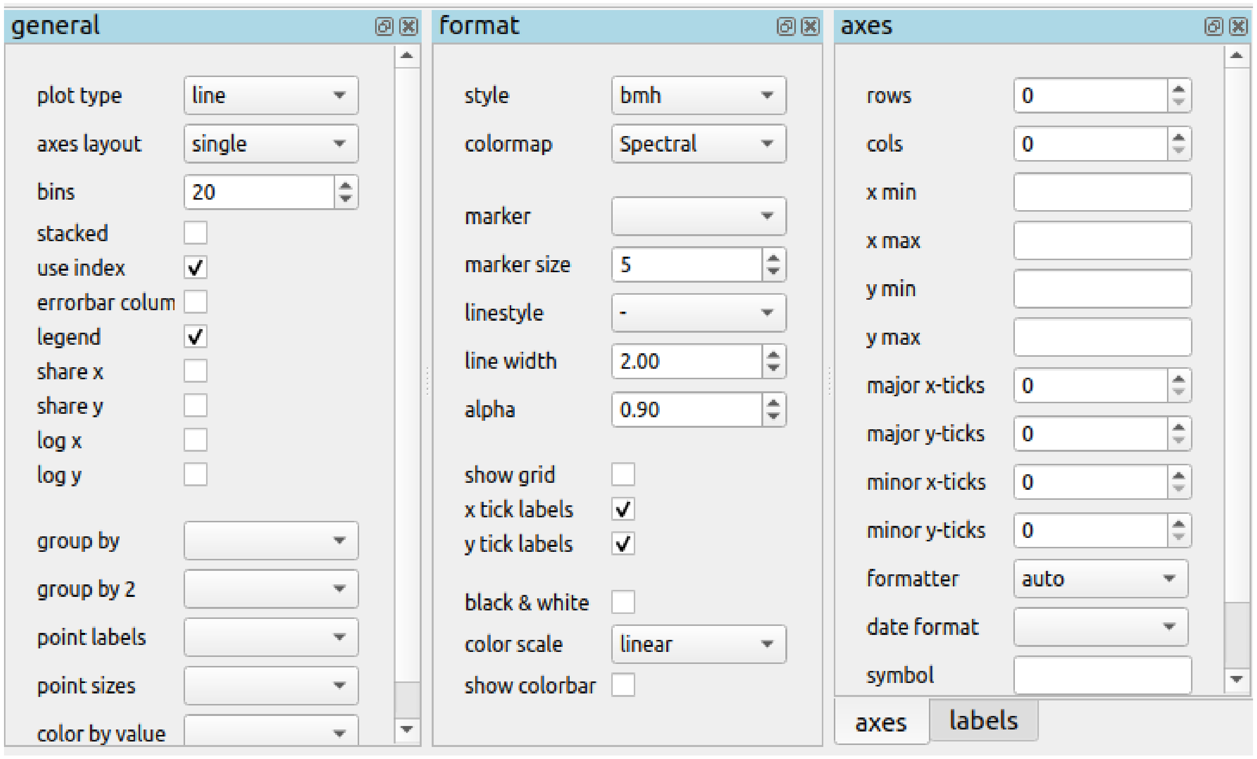
It may sometimes be necessary to re-format column names, for example to remove unwanted characters. If you have dozens or more columns this would be time consuming, so there is a function in tablexplore to do this in one step. Accessed from Tools->Convert column names, this dialog allows you to replace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

6.16 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 we create a date column and then use resampling to smooth out the data in another column using a window of 7.

6.17 Plot options

The plot options are a series of docked widgets on the right side of the plot, grouped by functionality. The docks can be dragged to the other sides of the application window or closed. Re-opening is done from the dock menu. Most default formatting options such as the type of plot, whether to show a legend etc. are in the 'general' tab. If you use the program regularly you will be familiar with where things are.



plot type	line
axes layout	single
bins	20
stacked	☐
use index	✓
errorbar colum	☐
legend	✓
share x	☐
share y	☐
log x	☐
log y	☐
group by	
group by 2	
point labels	
point sizes	
color by value	

style	bmh
colormap	Spectral
marker	
marker size	5
linestyle	-
line width	2.00
alpha	0.90
show grid	☐
x tick labels	✓
y tick labels	✓
black & white	☐
color scale	linear
show colorbar	☐

rows	0
cols	0
x min	
x max	
y min	
y max	
major x-ticks	0
major y-ticks	0
minor x-ticks	0
minor y-ticks	0
formatter	auto
date format	
symbol	

The following plot types are currently supported:

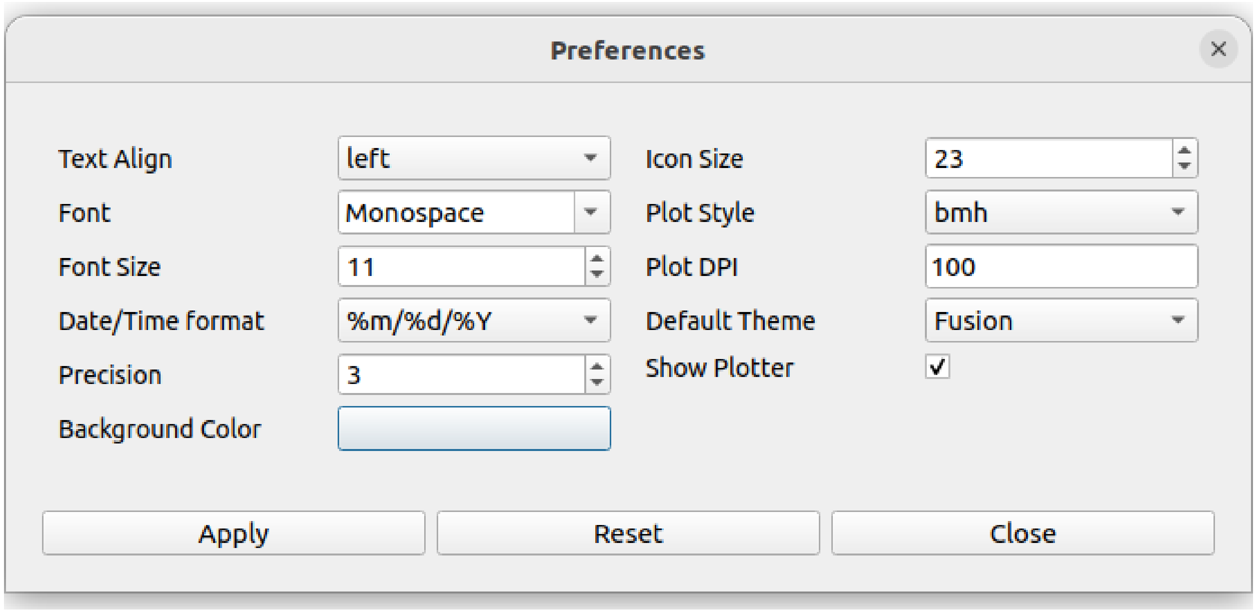
- line
- bar
- barh
- scatter
- pie
- histogram
- box plot
- dot plot
- heatmap
- area
- hexbin
- contour
- scatter matrix

6.18 Plotting grouped data

Rather than grouping the table directly and then plotting, 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 1-2 columns at once.

6.19 Setting preferences

Application settings are set from the Edit->Preferences menu. The image below shows the settings which are mostly self explanatory. If settings get corrupted or you want to restore defaults use the 'reset' button.



Text Align	left	Icon Size	23
Font	Monospace	Plot Style	bmh
Font Size	11	Plot DPI	100
Date/Time format	%m/%d/%Y	Default Theme	Fusion
Precision	3	Show Plotter	✓
Background Color			

6.20 Plugins

Plugins can be added by anyone (see code examples on how to do this). Currently there are are only a few useful built-in plugins. New ones will be added below. To add a third party plugin (just a .py file), place it in the plugin folder under <home dir>/.config/tablexplore. For security, you shouldn't just download and run any .py file without trusting it first.

6.20.1 Colormap plugin

This allows you to add your own colormaps for plotting. The screen grab below shows you. You can generate random colors, then edit them. When done choose the type of colormap and then save. Pick a name and this is stored and added to the list of of colormaps in the plot options. You have to restart the program to see it. (Colormaps are kept under .config/tablexplore/cmaps.pkl which can be deleted if you want to clear them.)

CHAPTER

SEVEN

CODE EXAMPLES

This section is for Python programmers.

7.1 Basics

If you want to use the table widget in another GUI program:

```
python
from PySide2 import QtCore
from PySide2.QtWidgets import *
from PySide2.QtGui import *
import pandas as pd
from tablexplore import data, core, plotting, interpreter

class TestApp(QMainWindow):
    def __init__(self, project_file=None, csv_file=None):

        QMainWindow.__init__(self)
        self.setAttribute(QtCore.Qt.WA_DeleteOnClose)
        self.setWindowTitle("Example")
        self.setGeometry(QtCore.QRect(200, 200, 800, 600))
        self.main = QWidget()
        self.setCentralWidget(self.main)
        layout = QVBoxLayout(self.main)
        df = data.getSampleData()
        t = core.DataFrameWidget(self.main,dataframe=df)
        layout.addWidget(t)
        #show a Python interpreter
        t.showInterpreter()
        return

if __name__ == '__main__':
    import sys
    app = QApplication(sys.argv)
    aw = TestApp()
    aw.show()
    app.exec_()
```

7.2 Writing a plugin

This is quite straightforward if you are familiar with PyQt5/Pyside2. Built in plugins are kept in the plugin folder where the program is installed. You can look at these to get an idea how the plugins are written. When you make your own plugin, just add the .py file to the plugin folder under <home dir>/.config/tablexplore. It will be loaded when the program starts and added to the menu. You can add any code into the script, usually designed to execute using the table or plotter. GUI based plugins will be added as docked widgets to the application.

Here is an example plugin:

```
class ExamplePlugin(Plugin):
    """Template plugin for TableExplore"""

    #uncomment capabilities list to appear in menu
                capabilities = ['gui']
    requires = ['']
    menuentry = 'Example Plugin'
    name = 'Example Plugin'

    def __init__(self, parent=None, table=None):
        """Customise this and/or doFrame for your widgets"""

        if parent==None:
            return
        self.parent = parent
        self.table = table
        self.createWidgets()
        return
```

CHAPTER

EIGHT

TABLEXPLORE

8.1 tablexplore package

8.1.1 Submodules

8.1.2 tablexplore.app module

8.1.3 tablexplore.core module

8.1.4 tablexplore.dialogs module

8.1.5 tablexplore.interpreter module

8.1.6 tablexplore.plotting module

8.1.7 tablexplore.plugin module

8.1.8 tablexplore.qt module

8.1.9 tablexplore.terminal module

8.1.10 tablexplore.util module

8.1.11 tablexplore.widgets module

8.1.12 Module contents

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search