Journal of open research software

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

SOFTWARE METAPAPER

# DataExplore: An Application for General Data Analysis in Research and Education

Damien Farrell[1]

[1] UCD School of Veterinary Medicine, University College Dublin, Ireland
farrell.damien@gmail.com

DataExplore is an open source desktop application for data analysis and plotting intended for use in both research and education. It is intended primarily for non-programmers who need to do relatively advanced table manipulation methods. Common tasks that might not be familiar to spreadsheet users such as table pivot, merge and join functionality are included as core elements. Creation of new columns using arithmetic expressions and pre-defined functions is possible. Table filtering may be done with simple boolean queries. The other primary feature is rapid dynamic plot creation from selected data. Multiple plots from various selections and data sources can also be composed using a grid layout. It is thus possible to create publication quality plots. A plugin system allows the addition of features with several plugins already available by default. The program is written in Python and is based on the PyData suite of Python libraries.

**Keywords:** python; scientific plotting; table analysis; pandas
**Funding statement:** The author is funded by an Irish Research Council Postdoctoral Fellowship (GOIPD/2015/475).

## (1) Overview

### Introduction

Recent years have seen a rapid growth in the importance of data handling and analysis in the sciences. Such is the complexity and volume of data it has given rise to data scientist specialisations within many fields. In the biological sciences, the data analysis task is often assigned to a bioinformatician. Such expert skills are essential when the complexity of the task is too much for experimentalists unfamiliar with advanced computational techniques. However there is a danger of over reliance upon data analysts particularly in cases where an analysis can be done with relatively basic computational skills.

Spreadsheets are widely used in scientific research. They have also advanced a great deal in sophistication [1] since their introduction and are by now a standard tool for anyone dealing with numerical data. In the sciences there has however been a tendency to rely on the spreadsheet for tasks that they were not originally designed for [2]. Though advanced features like pivot tables are available many general users are sometimes not aware of them and make the worksheet more complicated than it needs to be. Even if a spreadsheet can perform a task using a macro it is often much more complex to accomplish than it would be with a few lines of code [3]. Spreadsheets have another more serious problem in that they make reproducible analysis very difficult. This arises out of multiple factors. The opaque way in which cell based formulae are often used makes it hard to track calculations. The use of conditional formatting to present results makes it almost impossible to interpret them in another format. Also because they may be used for data entry and analysis at the same time, *ad hoc* changes to the raw data are encouraged. Finally, statistical analysis using the most common commercial product, Excel, have been criticized [4]. These problems make reproducible science difficult.

For the general scientific user these limitations can be partly overcome by using other tools to compliment spreadsheets. Many researchers use separate plotting packages such as GraphPad Prism [5] and statistical tools like SPSS [6] for analysis. However this gives rise to another problem – these are commercial applications. They are very expensive and the source code is closed. This has serious implications for reproducibility. The tendency to use commercial software is highly prevalent in academia even though there are some viable open source solutions available. This may partly be a problem of general awareness on the part of the user. Veusz [7] and SciDAVis [8] are good examples of free plotting packages that compare favourably with commercial products though they do not seem to be widely known. Commercial tools are also frequently rather feature heavy and complex for the general user with entire courses devoted to teaching them.

Scientists in certain data intensive subject areas, are now beginning to adapt to scripting languages like R and Python [9]. These are a much better foundation to build future skills on and since they are open platforms they allow users to publish full end-to-end instructions that anyone in the world can reproduce for free. They also facilitate workflows with large data [10]. Adoption of these scripting tools is easier said than done because of the intimidating nature of programming to many. This is one reason R might not easily gain traction with experimentalists since it requires at least some programming skill. R-studio [11] goes a long way to address this issue as it provides a user friendly environment for newer users. Though much progress has been on new web based tools it still a challenge to build highly interactive applications inside the browser. There is still therefore space for graphical desktop tools to provide a familiar compliment to spreadsheets for non-programmers.

### *Objectives*

*DataExplore* is intended for rapid exploratory analysis of tabulated data. Quick transformation and visualization of data are core features. The use of the Python PyData stack [12] as the back-end means a large number of well tested algorithms are already available. The dichotomy between programming and tools with a graphical interface is usually a sharp one [13] with users preferring one or the other approach. *DataExplore* is also intended to help bridge this gap by readily making possible processing steps normally familiar to data analysts. The main objectives of the software are:

- allow quick exploration and visualization of a data set
- allow a familiar graphical interface but implement more advanced table analysis features than currently accessible in spreadsheets
- help to bridge the gap between graphical interface and command driven or programmatic approaches to data analysis
- scale to medium sized datasets, i.e. a table of the order of 1–5 million rows that will fit in the memory of most computers
- allow publication quality plots to be made easily and encourage clear scientific visualization [14]

## Implementation and architecture

### *Methodology*

The core R data structure is called data.frame, a versatile matrix structure that stores multiple data types [15]. It has been replicated in Python as a core component of the Pandas library [16]. This has opened the way to much more convenient R style data analysis in Python. Pandas DataFrame structures, which use the efficient ndarray data container class in numpy [17], are now well integrated into other Python data analysis libraries, creating a very useful ecosystem. These libraries are often grouped together as part of the PyData stack [18]. DataExplore is based on using DataFrames to present tabulated data and on matplotlib [19] for plotting. Matplotlib is a very well established plotting library for Python and produces publication-quality figures in a variety of hard copy formats and interactive environments across platforms.

In some plotting packages like Veusz, SciDaviz and mjograph [20] plots are designed by the addition of multiple plot elements to which data is attached. DataExplore is more data centric like R-studio and plots are generated dynamically from the currently selected data and chosen options. The idea is that rows and columns can quickly be chosen or tables edited and new plots generated instantly with minimal mouse clicks. Plots cannot currently be interactively edited though this is an option that could be added later.

### *Architecture*

The software is written in Python and makes extensive use of the PyData libraries. These form an ecosystem of libraries that can provide a complete solution to data analysis from visualization to machine learning. The graphical interface is built with Tkinter/ttk, the standard graphical tool kit for Python. Like other Python packages the library is broken into modules which contain classes grouped by function. Table, plotting and dialog widgets are in their own modules as shown in **Figure 1**. The core class is the pandastable widget which is a Tkinter canvas object. This is used to display a Pandas DataFrame via a model class that carries out changes to the DataFrame based on user interaction and stores some additional data about the Table. This widget is designed to be re-used in any Tkinter application. The DataExplore application module itself is built around the table widget, a plot viewer module and several plugins.

### *User interface*

The application consists essentially of a table and associated plot viewer, shown in **Figure 2**. Multiple sets of tables can be loaded and saved as single projects. For certain functions a child table or sub-table is created below the main one. This may be to store the results of a table manipulation such as an aggregation or to paste in another table so that it can be joined to the main one. Another use would be to paste a portion of the selected data and plot it. The sub-table can be created and discarded as needed.

Unlike a spreadsheet, the focus is not on data entry. Though individual cell entry is possible, users are encouraged to keep their original data separate and unchanged. Results can be exported to csv or other formats if required. This is important to robust analysis. An undo/redo feature is not yet implemented but will likely be useful in the future when more complex series of processing steps need to be experimented with.

Plot options are laid out in a set of tabbed control panels below the plot allowing the user to switch quickly between basic and other plotting modes. Currently a 3D plot mode and grid layout options are also available. Table functions are accessed either from the right toolbar (see **Figure 2**), the right-click context menu inside the table or from the main menu. Dialogs such as plugin interfaces are usually placed below the table.

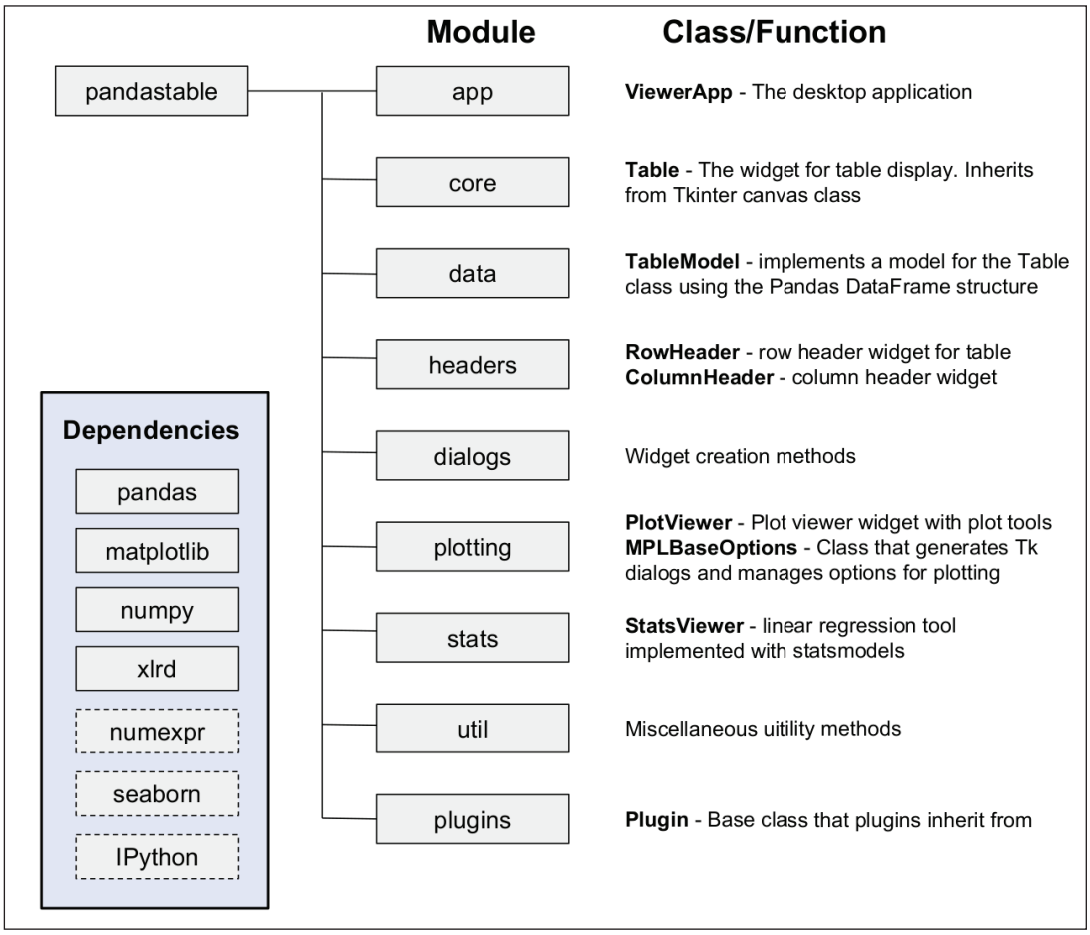


**Figure 1:** Outline of pandastable library modules. The graphical user interface is a scaffolding using these modules.

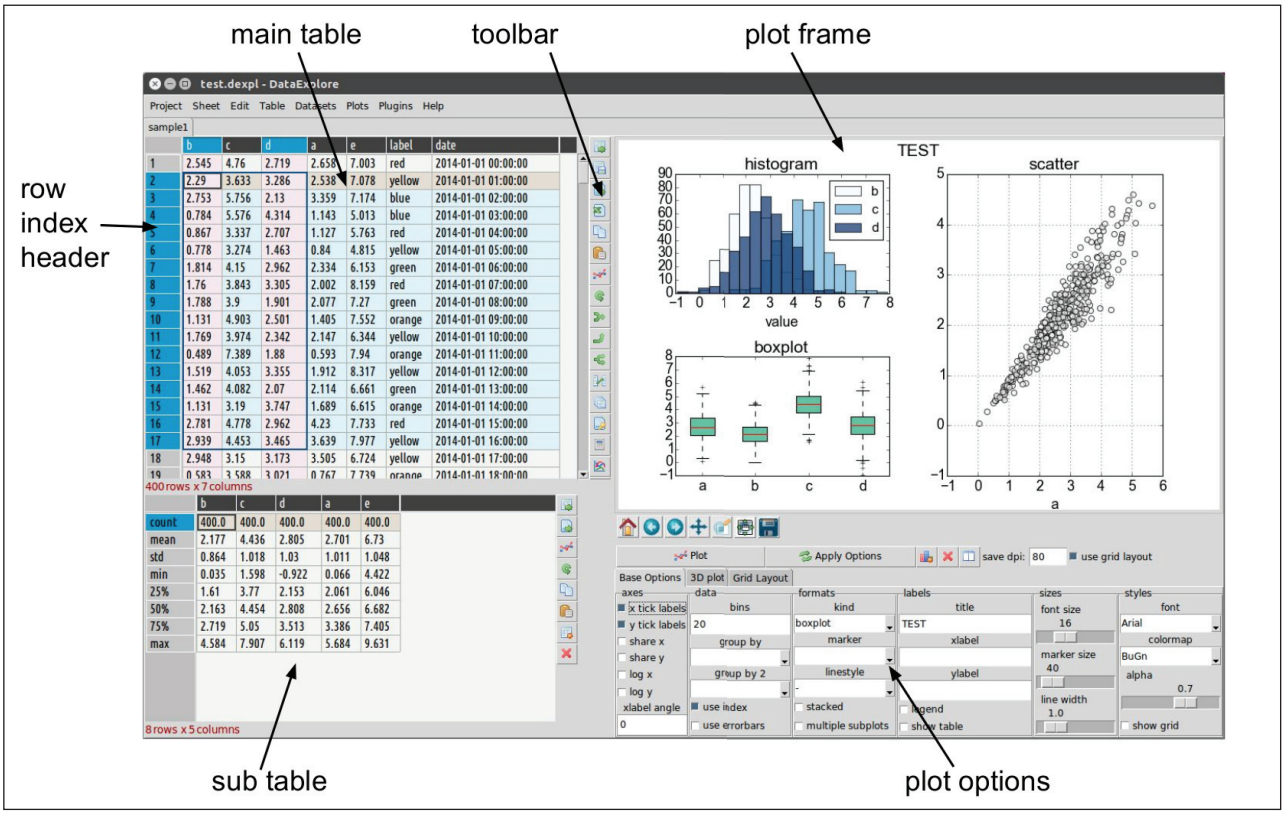


**Figure 2:** Application interface.

### *Features*

#### Import of text files

Import of csv and general plain text formats is a standard feature of Pandas using the read_csv method and supports many options. The most essential of these are available via the import dialog accessible from the toolbar or by right-clicking anywhere in the table and using the context menu.

#### Row and column indexes

The index is a fundamental feature of the underlying DataFrame. This performs the central role of data alignment or getting and setting of subsets of the table. A more novel aspect is the use of "hierarchical" indexing. This is essentially a way of representing data with an arbitrary number of dimensions in a 2D table. In our program mostly the use of multi-indexes is implicit to the way the program works but it opens the door to add more useful functionality later on. For now the index can be displayed or hidden in the table and columns can be turned into indexes. This is useful for plotting since the index is often the implied x-axis for plotting.

#### Table filtering

Currently filtering of the table is done using a quite simple string query method. An entry box is used to enter the query and the table updated accordingly. The main table is restored when the filters are cleared. The syntax is straightforward to learn for beginners and may be useful for teaching logical AND/OR/NOT row-wise operations.

#### Table manipulation

Common transformations such as transpose, aggregation, pivot and merge are supported. Results are mostly placed in the sub-table so as not to overwrite the main table. The sub-table can also be used to plot from or copied into the main table or another sheet. For operations involving two tables (like concatenate or merge) the second dataset is loaded into the sub-table (by importing or pasting) which can then be joined to the main table.

#### Plotting behaviour

The design is oriented around quick generation of plots from the current selections. This means that the current plot is constantly overridden. However plots can be saved and recalled in the current session if required. To produce multiple plots in one figure a grid layout mode is used. Changing the number of rows/columns makes a finer grid and adjusting the row/column spans allows a variety of sub plot combinations to be created. When the user wants to add a new sub plot they simple select the row and column location to add them. An example is shown in **Figure 3**. It is also possible to use this method to make inset plots by overlaying them on the main plot.

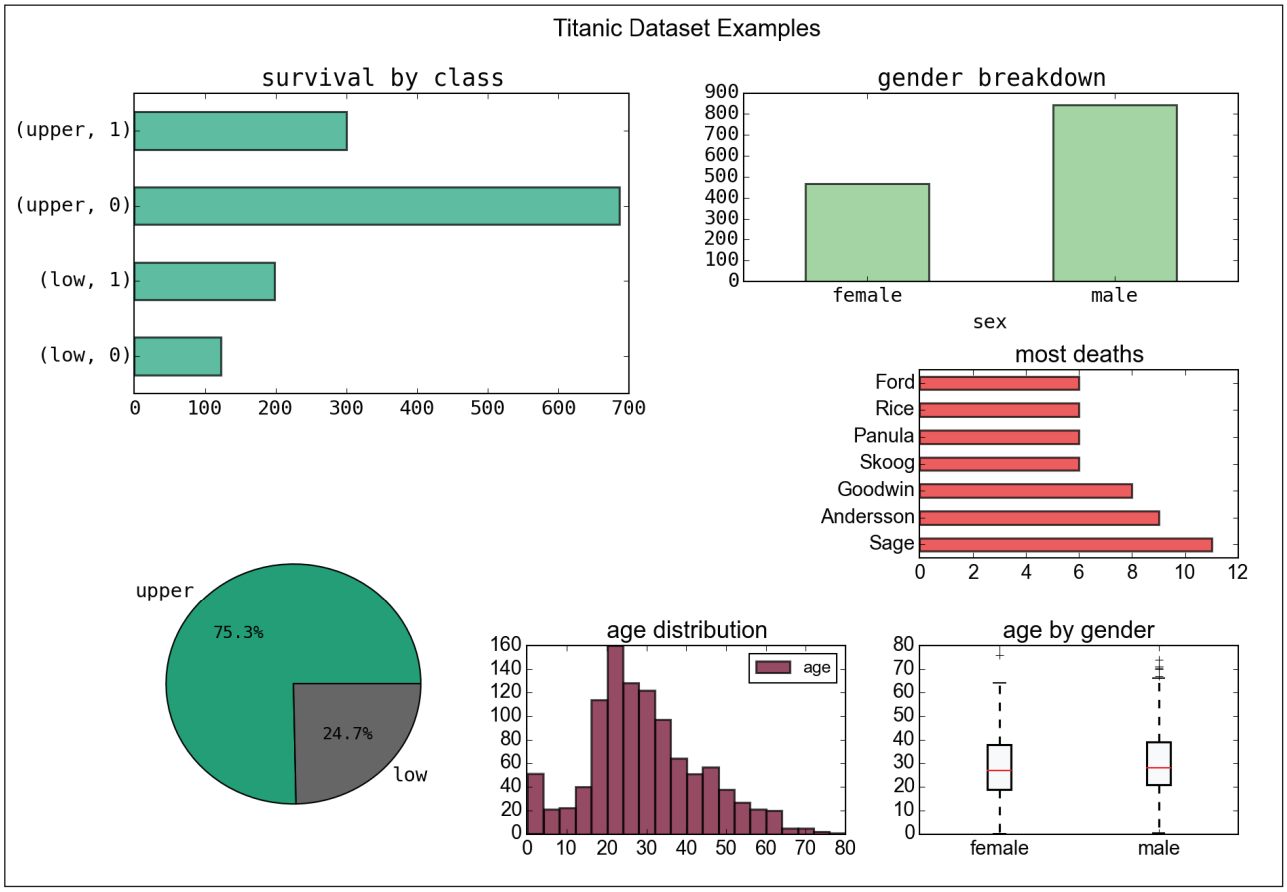


**Figure 3:** A figure generated directly from the application using the Titanic data. This uses the grid layout mode to combine multiple sub plots together.

#### Categorical plots

Data can be grouped and plotted either by grouping by categorical columns in the plot dialog or performing a groupby-aggregate step and plotting the resulting table. The factor plots plugin provides even more advanced plotting capabilities. Factor plots allow multiple comparisons to be made in a single graph. That is, you can split data by more than one variable along an axis or between plots. In seaborn these dimensions are called row/col (the plot dimensions) x,y (axes) and hue (grouping/color within plots). These concepts are illustrated in the seaborn documentation on factor plotting [21] and on the blog.

#### 3D plots

Plotting interactive 3D projections that can be zoomed and rotated is provided using the matplotlib mplot3d module. Scatter, bar, contour, wireframe and surface plots are available. Plotting of column selections does not have a single unambiguous representation in 3D space for the latter three kinds of plots. So pre-defined modes must be used that tell the program how to interpret the selected data. Currently the default is to interpret the third column $z$ as a function of the first two $x$ and $y$, i.e. of the form $(x,y) \rightarrow z$. Other modes, such as support for parameterized functions, are still to be added.

#### Data fitting

The statsmodels [22] library is used for data fitting in DataExplore since it works well with Pandas and has a simple programming interface. It provides descriptive statistics, statistical tests, plotting functions, and implementation of standard estimators used in model fitting. String formulas are supported using Patsy and this is used in to allow the user to type in their formulas providing using the special syntax.

#### Plugins

Plugins are for adding custom functionality that is not present in the main application. These are Python scripts implemented by sub-classing the Plugin class in the plugin module. At minimum a plugin must have a main() method which is called by the application to launch it. Otherwise a script can generally contain any code the author wishes. Usually the idea will be to implement a dialog that the user interacts with but this could also be a single function that runs on the current table or all sheets at once without further user interaction. Three plugins are currently provided with the library:

- Batch file renaming utility – a tool for renaming multiple files at once that can be useful for importing files.
- Factor plotting – advanced categorical plotting using the Seaborn library [23]. seaborn provides a high-level interface to matplotlib for drawing attractive graphics. It also understands DataFrames without any need for converting them.
- IPython console – Here you can call any Python commands and even shell commands provided in IPython [24]. The underlying table DataFrame can be manipulated directly with code snippets or external scripts and the table updated to reflect the changes immediately. This should prove a useful way to teach coding skills in a familiar environment.

### *Documentation and usage*

Documentation is provided in the form of a wiki on github at https://github.com/dmnfarrell/pandastable/wiki/. Specific case studies/tutorials along with links to screen casts and details of new features can be viewed on the blog at http://dmnfarrell.github.io/. The case studies provide a visual guide through real world examples. This blog will be kept up to date as the program is further developed.

#### Current case studies

1. Looking at the Titanic dataset (basic). Exploratory methods for beginners using the Titanic data from a Kaggle Getting Started Competition [25]. This illustrates the used of the software in initial explorations of data for beginners using plots of distributions, breaking down columns by category and re-binning categorical data.
2. Plotting miRNA abundance data (advanced). This uses miRNA-sequencing expression data [26] to show more complex methods of representing data sets with multiple sample labels. It includes a demonstration of how to create long form data for use with the factor plotting plugin.

### *Future developments*

The software at time of writing is at version 0.7. Regular releases will be made as bugs are fixed or features added. Many useful developments can be applied to this software to exploit the wealth of functionality in Python scientific libraries. The plugin system allows such new features to be added very easily by third parties with some knowledge of Python. It is important to underline that development should be led by user feedback rather than the authors. Some particularly useful potential features are mentioned here:

- Workflow tracking mechanism. Obviously in a point and click environment people will not remember every step they take in an analysis. To aid reproducibility some method of recording processing steps would be useful.
- Integration with Jupyter notebooks. The Jupyter notebook [27] is a web application that allows you to create and share documents that contain live code, visualizations and explanatory text. It has potential to be used for the kind of workflow tracking mentioned above.
- Plugins for more specific kinds of data analysis such as principal component analysis.
- Batch conversion and/or joining of multiple text/csv files likely using a plugin.
- Improvements to scale with larger data sets.
- Add support for loading remote data sources and sharing results.
- Enhance plot support with the ability to add annotations and allow arbitrary placement of sub plots amongst other features.

### Quality control
Git is used for version control and bug tracking. Github's issue tracking supports milestones, labels and assignees for filtering and categorizing bugs. It is therefore the ideal way to handle user feedback.

#### *Testing*
Testing is done using the standard Python unittest framework (PyUnit). Tests can be executed from the cloned source directory. Since automated testing of Tkinter widgets is known to be difficult, the tests currently concentrate on table functions that do not require user interaction. For other tests user feedback is essential. The project uses the Travis continuous integration service [28]. Travis CI automatically detects when a commit has been made and pushed to the GitHub repository. Each time this happens, it will try to build the project and run the tests. This model of development scales well for multiple developers.

#### *Data integrity*
Since there is minimal emphasis on data entry per se, the user is encouraged to keep raw data unchanged. Projects saved in the native format as multiple sets of worksheets are kept separate from the original data. Projects are saved in MessagePack format [29] which is an efficient binary serialization format and is used to save Python objects. In our case the DataFrame, meta data like the current table selections and plotting options (as Python dictionaries) for each sheet are saved together in one project so that the workspace can be reloaded conveniently. Individual DataFrames can also be saved alone without any extra meta data so that they can be persisted and reloaded outside the context of the application. Though we have found the format efficient and reliable, MessagePack support is still experimental in Pandas. The project files are not designed be used as an archive for raw data sets but rather seen as a workspace that can be updated constantly and interchanged between users wishing to share analysis and workflows. A methodology for workflow tracking is planned that will involve refinement of the this file format.

## (2) Availability
### Operating system
This software is supported on any operating system that supports a standard Python installation which includes Linux, Windows and OSX. In all systems the DataExplore application can be provided by installing the pandastable Python library via pip or easy_install. This requires a working Python installation. It is also available as a package via the self contained Anaconda Python distribution. In Windows a binary installer is available, packaged with cx_Freeze [30], that installs an executable and all the required libraries without the need for a separate Python install. This installer is a 32-bit executable which will run on all windows systems. Detailed install instructions are given in the documentation.

### Programming language
Python version >= 3.4 or 2.7.

### Additional system requirements
No special requirements.

### Dependencies
The following Python libraries are required dependencies:
Numpy >= 1.5
matplotlib >= 1.1
pandas >= 0.17
numexpr >= 2.4
xlrd >= 0.9

### Optional dependencies
seaborn >= 0.6
statsmodels >= 0.6 (requires scipy)
ipython >= 4.0

### Software location
***Archive*** (e.g. institutional repository, general repository)
***Name:*** Zenodo
***Persistent identifier:*** 10.5281/zenodo.44891
***Licence:*** GPL v3
***Publisher:*** Damien Farrell
***Date published:*** *17/1/16*
***Code repository***
***Name:*** GitHub
***Identifier:*** https://github.com/dmnfarrell/pandastable
***Licence:*** GPL v3
***Date published:*** 11/2/14

### Language
English.

## (3) Reuse potential
The software is designed for a general science and technical audience and is not tied to any specific field. Though it was initially developed with the biological sciences in mind, it has very general application. It will be most useful for students and researchers who are not programmers but need to do convenient exploration of their data. Educators at all levels are also a target for the software. With the easy to learn user interface it is a good way to introduce basic data manipulation methods. Specific uses include producing plots for reports or publication, quick visualization of small to medium sized datasets, database style filtering with string queries or fitting linear models. It is hoped that some of this functionality will help students become familiar with the more advanced analytical methods available via programming languages. Data scientists may also find the tool useful for quick plots of their data before or after detailed analyses and as a way of sharing results with others.

The use of proprietary software without access to the code base is still very common in science. This makes it hard to do reproducible work that can be shared. This software is based on well established open source Python libraries that do not have such issues. These high-quality tools for scientific computing provide the ideal platform to build a user friendly open source application for data analysis. In the long-term, it is hoped that a community of users can be built, some of whom will be developers able to

provide their own plugins or extend the core application. The project can be also be forked without restriction.

### Competing Interests

The authors declare that they have no competing interests.

### Acknowledgements

Thanks to Prof. Stephen Gordon for supporting work on this project.

### References

1. **Weathington, J** 2015 5 things every data scientist should know about Excel. Available at http://www.techrepublic.com/article/5-things-every-data-scientist-should-know-about-excel/ [Accessed: 16-Jan-2016].
2. **Burns, P** 2014 Spreadsheet Addiction. Available at http://www.burns-stat.com/documents/tutorials/spreadsheet-addiction/.
3. **Moffitt, C** 2014 Common Excel Tasks Demonstrated in Pandas. Available at http://pbpython.com/excel-pandas-comp.html.
4. **McCullough, B D** and **Heiser, D A** 2008 On the accuracy of statistical procedures in Microsoft Excel 2007. *Comput Stat Data Anal*, 52(10): 4570–4578. DOI: http://dx.doi.org/10.1016/j.csda.2008.03.004
5. **GraphPad Software** 2015 GraphPad Prism version 6.0. Available at http://www.graphpad.com/scientific-software/prism/.
6. **IBM** 2015 IBM SPSS Statistics. Available at http://www-01.ibm.com/software/analytics/spss/.
7. **Sanders, J** 2015 Veusz. Available at http://home.gna.org/veusz/.
8. **Benkert, T, Franke, K** and **Standish, R** 2007 SciDAVis. Available at http://scidavis.sourceforge.net/.
9. **Buffalo, V** 2015 *Bioinformatics Data Skills*. O'Reilly Media.
10. **Heller, M** 2015 Learn to crunch big data with R. Available at http://www.infoworld.com/article/2880360/big-data/learn-to-crunch-big-data-with-r.html [Accessed: 07-Jan-2016].
11. **RStudio Inc.** 2015 RStudio: Integrated Development for R. Boston MA.
12. **Oliphant, T E** 2007 (May) Python for Scientific Computing. *Comput Sci Eng*, (9)3: 10–20. DOI: http://dx.doi.org/10.1109/MCSE.2007.58
13. **Ward, N** 2013 Excel, SPSS, Minitab or R? Available at https://learnandteachstatistics.wordpress.com/2013/02/11/excel-spss-minitab-or-r/ [Accessed: 09-Sep-2015].
14. **Rougier, N P, Droettboom, M** and **Bourne, P E** 2014 Ten Simple Rules for Better Figures. *PLoS Comput Biol*, 10(9): e1003833. DOI: http://dx.doi.org/10.1371/journal.pcbi.1003833
15. **Data science retreat** 2013 R: the good parts. Available at http://blog.datascienceretreat.com/post/69789735503/r-the-good-parts [Accessed: 08-Jan-2016].
16. **Mckinney, W** 2015 Pandas, Python Data Analysis Library. Available at http://pandas.pydata.org/.
17. **van der Walt, S, Colbert, S C** and **Varoquaux, G** 2011 (March) The NumPy Array: A Structure for Efficient Numerical Computation. *Comput Sci Eng*, 13(2): 22–30. DOI: http://dx.doi.org/10.1109/MCSE.2011.37
18. **The PyData Community** 2015 PyData. Available at http://pydata.org/downloads/.
19. **Hunter, J D** 2007 (May) Matplotlib: A 2D Graphics Environment. *Comput Sci Eng*, 9(3): 90–95. DOI: http://dx.doi.org/10.1109/MCSE.2007.55
20. **Tanahashi, M** 2014 mjograph. Available at http://www.ochiailab.dnj.ynu.ac.jp/mjograph/.
21. **Waskom, M** 2015 Seaborn factorplot documentation. Available at http://stanford.edu/~mwaskom/software/seaborn/generated/seaborn.factorplot.html [Accessed: 12-Jan-2016].
22. **Statsmodels Developers** 2015 Statsmodels. Available at http://statsmodels.sourceforge.net/.
23. **Waskom, M** 2012 Seaborn. Available at http://stanford.edu/~mwaskom/software/seaborn/. DOI: http://dx.doi.org/10.1109/MCSE.2007.53
24. **Pérez, F** and **Granger, B E** 2007 (May) {IP}ython: a System for Interactive Scientific Computing. *Comput Sci Eng*, 9(3): 21–29.
25. **Kaggle** 2012 Titanic: Machine Learning from Disaster. Available at https://www.kaggle.com/c/titanic [Accessed: 16-Jan-2016].
26. **Farrell, D, Shaughnessy, R G, Britton, L, MacHugh, D E, Markey, B** and **Gordon, S V** 2015 The Identification of Circulating MiRNA in Bovine Serum and Their Potential as Novel Biomarkers of Early Mycobacterium avium subsp paratuberculosis Infection. *PLoS One*, 10(7): e0134310. DOI: http://dx.doi.org/10.1371/journal.pone.0134310
27. **Project Jupyter** 2015 Jupyter Notebook. Available at http://jupyter.org/ [Accessed: 21-Jan-2016].
28. **Travis CI Community** 2011 Travis continuous integration. Available at https://travis-ci.org/.
29. **Furuhashi, S** 2008 MessagePack. Available at http://msgpack.org/index.html.
30. **Tuininga, A** 2014 cx_Freeze. Available at http://cx-freeze.sourceforge.net/.

**How to cite this 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

**Submitted:** 15 September 2015 **Accepted:** 08 March 2016 **Published:** 22 March 2016



]u[ *Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS