pandas: powerful Python data analysis toolkit

Release 1.0.5

Wes McKinney and the Pandas Development Team

Jun 17, 2020

CONTENTS

Date: Jun 17, 2020 **Version**: 1.0.5

Download documentation: PDF Version | Zipped HTML

Useful links: Binary Installers | Source Repository | Issues & Ideas | Q&A Support | Mailing List

`pandas` is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

CHAPTER

ONE

GETTING STARTED

1.1 Installation

Before you can use pandas, you'll need to get it installed.

Pandas is part of the Anaconda distribution and can be installed with Anaconda or Miniconda:

```
conda install pandas
```

Pandas can be installed via pip from PyPI.

```
pip install pandas
```

Learn more

1.2 Intro to pandas

Straight to tutorial...

When working with tabular data, such as data stored in spreadsheets or databases, Pandas is the right tool for you. Pandas will help you to explore, clean and process your data. In Pandas, a data table is called a `DataFrame`.

To introduction tutorial

To user guide

Straight to tutorial...

Pandas supports the integration with many file formats or data sources out of the box (csv, excel, sql, json, parquet,...). Importing data from each of these data sources is provided by function with the prefix `read_*`. Similarly, the `to_*` methods are used to store data.

To introduction tutorial

To user guide

Straight to tutorial...

Selecting or filtering specific rows and/or columns? Filtering the data on a condition? Methods for slicing, selecting, and extracting the data you need are available in Pandas.

To introduction tutorial

To user guide

Straight to tutorial...

Pandas provides plotting your data out of the box, using the power of Matplotlib. You can pick the plot type (scatter, bar, boxplot,...) corresponding to your data.

To introduction tutorial

To user guide

Straight to tutorial...

There is no need to loop over all rows of your data table to do calculations. Data manipulations on a column work elementwise. Adding a column to a `DataFrame` based on existing data in other columns is straightforward.

To introduction tutorial

To user guide

Straight to tutorial...

Basic statistics (mean, median, min, max, counts...) are easily calculable. These or custom aggregations can be applied on the entire data set, a sliding window of the data or grouped by categories. The latter is also known as the split-apply-combine approach.

To introduction tutorial

To user guide

Straight to tutorial...

Change the structure of your data table in multiple ways. You can `melt()` your data table from wide to long/tidy form or `pivot()` from long to wide format. With aggregations built-in, a pivot table is created with a sinlge command.

To introduction tutorial

To user guide

Straight to tutorial...

Multiple tables can be concatenated both column wise as row wise and database-like join/merge operations are provided to combine multiple tables of data.

To introduction tutorial

To user guide

Straight to tutorial...

Pandas has great support for time series and has an extensive set of tools for working with dates, times, and time-indexed data.

To introduction tutorial

To user guide

Straight to tutorial...

Data sets do not only contain numerical data. Pandas provides a wide range of functions to cleaning textual data and extract useful information from it.

To introduction tutorial

To user guide

1.3 Coming from...

Currently working with other software for data manipulation in a tabular format? You're probably familiar to typical data operations and know *what* to do with your tabular data, but lacking the syntax to execute these operations. Get to know the pandas syntax by looking for equivalents from the software you already know:

Learn more

Learn more

Learn more

Learn more

1.4 Community tutorials

The community produces a wide variety of tutorials available online. Some of the material is enlisted in the community contributed *Tutorials*.

1.4.1 Installation

The easiest way to install pandas is to install it as part of the Anaconda distribution, a cross platform distribution for data analysis and scientific computing. This is the recommended installation method for most users.

Instructions for installing from source, PyPI, ActivePython, various Linux distributions, or a development version are also provided.

Python version support

Officially Python 3.6.1 and above, 3.7, and 3.8.

Installing pandas

Installing with Anaconda

Installing pandas and the rest of the NumPy and SciPy stack can be a little difficult for inexperienced users.

The simplest way to install not only pandas, but Python and the most popular packages that make up the SciPy stack (IPython, NumPy, Matplotlib, ...) is with Anaconda, a cross-platform (Linux, Mac OS X, Windows) Python distribution for data analytics and scientific computing.

After running the installer, the user will have access to pandas and the rest of the SciPy stack without needing to install anything else, and without needing to wait for any software to be compiled.

Installation instructions for Anaconda can be found here.

A full list of the packages available as part of the Anaconda distribution can be found here.

Another advantage to installing Anaconda is that you don't need admin rights to install it. Anaconda can install in the user's home directory, which makes it trivial to delete Anaconda if you decide (just delete that folder).

Installing with Miniconda

The previous section outlined how to get pandas installed as part of the Anaconda distribution. However this approach means you will install well over one hundred packages and involves downloading the installer which is a few hundred megabytes in size.

If you want to have more control on which packages, or have a limited internet bandwidth, then installing pandas with Miniconda may be a better solution.

Conda is the package manager that the Anaconda distribution is built upon. It is a package manager that is both cross-platform and language agnostic (it can play a similar role to a pip and virtualenv combination).

Miniconda allows you to create a minimal self contained Python installation, and then use the Conda command to install additional packages.

First you will need Conda to be installed and downloading and running the Miniconda will do this for you. The installer can be found here

The next step is to create a new conda environment. A conda environment is like a virtualenv that allows you to specify a specific version of Python and set of libraries. Run the following commands from a terminal window:

```
conda create -n name_of_my_env python
```

This will create a minimal environment with only Python installed in it. To put your self inside this environment run:

```
source activate name_of_my_env
```

On Windows the command is:

```
activate name_of_my_env
```

The final step required is to install pandas. This can be done with the following command:

```
conda install pandas
```

To install a specific pandas version:

```
conda install pandas=0.20.3
```

To install other packages, IPython for example:

```
conda install ipython
```

To install the full Anaconda distribution:

```
conda install anaconda
```

If you need packages that are available to pip but not conda, then install pip, and then use pip to install those packages:

```
conda install pip
pip install django
```

Installing from PyPI

pandas can be installed via pip from PyPI.

```
pip install pandas
```

Installing with ActivePython

Installation instructions for ActivePython can be found here. Versions 2.7, 3.5 and 3.6 include pandas.

Installing using your Linux distribution's package manager.

The commands in this table will install pandas for Python 3 from your distribution. To install pandas for Python 2, you may need to use the `python-pandas` package.

Distribution	Status	Download / Repository Link	Install method
Debian	stable	official Debian repository	`sudo apt-get install python3-pandas`
Debian & Ubuntu	unstable (latest packages)	NeuroDebian	`sudo apt-get install python3-pandas`
Ubuntu	stable	official Ubuntu repository	`sudo apt-get install python3-pandas`
OpenSuse	stable	OpenSuse Repository	`zypper in python3-pandas`
Fedora	stable	official Fedora repository	`dnf install python3-pandas`
Centos/RHEL	stable	EPEL repository	`yum install python3-pandas`

However, the packages in the linux package managers are often a few versions behind, so to get the newest version of pandas, it's recommended to install using the `pip` or `conda` methods described above.

Installing from source

See the *contributing guide* for complete instructions on building from the git source tree. Further, see *creating a development environment* if you wish to create a *pandas* development environment.

Running the test suite

pandas is equipped with an exhaustive set of unit tests, covering about 97% of the code base as of this writing. To run it on your machine to verify that everything is working (and that you have all of the dependencies, soft and hard, installed), make sure you have pytest >= 5.0.1 and Hypothesis >= 3.58, then run:

```
>>> pd.test()
running: pytest --skip-slow --skip-network C:\Users\TP\Anaconda3\envs\py36\lib\site-
↪packages\pandas
============================= test session starts =============================
platform win32 -- Python 3.6.2, pytest-3.6.0, py-1.4.34, pluggy-0.4.0
```

(continues on next page)

(continued from previous page)

```
rootdir: C:\Users\TP\Documents\Python\pandasdev\pandas, inifile: setup.cfg
collected 12145 items / 3 skipped

..................................................................S......
........S.......................................................................
.......................................................................

==================== 12130 passed, 12 skipped in 368.339 seconds =====================
```

Dependencies

Package	Minimum supported version
setuptools	24.2.0
NumPy	1.13.3
python-dateutil	2.6.1
pytz	2017.2

Recommended dependencies

- numexpr: for accelerating certain numerical operations. `numexpr` uses multiple cores as well as smart chunking and caching to achieve large speedups. If installed, must be Version 2.6.2 or higher.
- bottleneck: for accelerating certain types of `nan` evaluations. `bottleneck` uses specialized cython routines to achieve large speedups. If installed, must be Version 1.2.1 or higher.

Note: You are highly encouraged to install these libraries, as they provide speed improvements, especially when working with large data sets.

Optional dependencies

Pandas has many optional dependencies that are only used for specific methods. For example, *`pandas.read_hdf()`* requires the `pytables` package, while *`DataFrame.to_markdown()`* requires the `tabulate` package. If the optional dependency is not installed, pandas will raise an `ImportError` when the method requiring that dependency is called.

Dependency	Minimum Version	Notes
BeautifulSoup4	4.6.0	HTML parser for read_html (see *note*)
Jinja2		Conditional formatting with DataFrame.style
PyQt4		Clipboard I/O
PyQt5		Clipboard I/O
PyTables	3.4.2	HDF5-based reading / writing
SQLAlchemy	1.1.4	SQL support for databases other than sqlite
SciPy	0.19.0	Miscellaneous statistical functions
XLsxWriter	0.9.8	Excel writing
blosc		Compression for HDF5
fastparquet	0.3.2	Parquet reading / writing

continues on next page

Table 1 – continued from previous page

Dependency	Minimum Version	Notes
gcsfs	0.2.2	Google Cloud Storage access
html5lib		HTML parser for read_html (see *note*)
lxml	3.8.0	HTML parser for read_html (see *note*)
matplotlib	2.2.2	Visualization
numba	0.46.0	Alternative execution engine for rolling operations
openpyxl	2.5.7	Reading / writing for xlsx files
pandas-gbq	0.8.0	Google Big Query access
psycopg2		PostgreSQL engine for sqlalchemy
pyarrow	0.12.0	Parquet, ORC (requires 0.13.0), and feather reading / writing
pymysql	0.7.11	MySQL engine for sqlalchemy
pyreadstat		SPSS files (.sav) reading
pytables	3.4.2	HDF5 reading / writing
pyxlsb	1.0.6	Reading for xlsb files
qtpy		Clipboard I/O
s3fs	0.3.0	Amazon S3 access
tabulate	0.8.3	Printing in Markdown-friendly format (see tabulate)
xarray	0.8.2	pandas-like API for N-dimensional data
xclip		Clipboard I/O on linux
xlrd	1.1.0	Excel reading
xlwt	1.2.0	Excel writing
xsel		Clipboard I/O on linux
zlib		Compression for HDF5

Optional dependencies for parsing HTML

One of the following combinations of libraries is needed to use the top-level `read_html()` function:

Changed in version 0.23.0.

- BeautifulSoup4 and html5lib
- BeautifulSoup4 and lxml
- BeautifulSoup4 and html5lib and lxml
- Only lxml, although see *HTML Table Parsing* for reasons as to why you should probably **not** take this approach.

Warning:

- if you install BeautifulSoup4 you must install either lxml or html5lib or both. `read_html()` will **not** work with *only* BeautifulSoup4 installed.
- You are highly encouraged to read *HTML Table Parsing gotchas*. It explains issues surrounding the installation and usage of the above three libraries.

1.4.2 Package overview

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible open source data analysis / manipulation tool available in any language**. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, `Series` (1-dimensional) and `DataFrame` (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, `DataFrame` provides everything that R's `data.frame` provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well:

- Easy handling of **missing data** (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be **inserted and deleted** from DataFrame and higher dimensional objects
- Automatic and explicit **data alignment**: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let *Series*, *DataFrame*, etc. automatically align the data for you in computations
- Powerful, flexible **group by** functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it **easy to convert** ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets
- Intuitive **merging** and **joining** data sets
- Flexible **reshaping** and pivoting of data sets
- **Hierarchical** labeling of axes (possible to have multiple labels per tick)
- Robust IO tools for loading data from **flat files** (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast **HDF5 format**
- **Time series**-specific functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging.

Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. pandas is the ideal tool for all of these tasks.

Some other notes

- pandas is **fast**. Many of the low-level algorithmic bits have been extensively tweaked in Cython code. However, as with anything else generalization usually sacrifices performance. So if you focus on one feature for your application you may be able to create a faster specialized tool.

- pandas is a dependency of statsmodels, making it an important part of the statistical computing ecosystem in Python.
- pandas has been used extensively in production in financial applications.

Data structures

Dimensions	Name	Description
1	Series	1D labeled homogeneously-typed array
2	DataFrame	General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column

Why more than one data structure?

The best way to think about the pandas data structures is as flexible containers for lower dimensional data. For example, DataFrame is a container for Series, and Series is a container for scalars. We would like to be able to insert and remove objects from these containers in a dictionary-like fashion.

Also, we would like sensible default behaviors for the common API functions which take into account the typical orientation of time series and cross-sectional data sets. When using ndarrays to store 2- and 3-dimensional data, a burden is placed on the user to consider the orientation of the data set when writing functions; axes are considered more or less equivalent (except when C- or Fortran-contiguousness matters for performance). In pandas, the axes are intended to lend more semantic meaning to the data; i.e., for a particular data set there is likely to be a "right" way to orient the data. The goal, then, is to reduce the amount of mental effort required to code up data transformations in downstream functions.

For example, with tabular data (DataFrame) it is more semantically helpful to think of the **index** (the rows) and the **columns** rather than axis 0 and axis 1. Iterating through the columns of the DataFrame thus results in more readable code:

```
for col in df.columns:
    series = df[col]
    # do something with series
```

Mutability and copying of data

All pandas data structures are value-mutable (the values they contain can be altered) but not always size-mutable. The length of a Series cannot be changed, but, for example, columns can be inserted into a DataFrame. However, the vast majority of methods produce new objects and leave the input data untouched. In general we like to **favor immutability** where sensible.

Getting support

The first stop for pandas issues and ideas is the Github Issue Tracker. If you have a general question, pandas community experts can answer through Stack Overflow.