Astronomical Data in Python

Astronomical Data in Python is an introduction to tools and practices for working with astronomical data. Topics covered include:

- Writing queries that select and download data from a database.
- Using data stored in an Astropy `Table` or Pandas `DataFrame`.
- Working with coordinates and other quantities with units.
- Storing data in various formats.
- Performing database join operations that combine data from multiple tables.
- Visualizing data and preparing publication-quality figures.

As a running example, we will replicate part of the analysis in a recent paper, "Off the beaten path: Gaia reveals GD-1 stars outside of the main stream" by Adrian M. Price-Whelan and Ana Bonaca.

This material was developed in collaboration with The Carpentries and the Astronomy Curriculum Development Committee, and supported by funding from the American Institute of Physics through the American Astronomical Society.

I am grateful for contributions from the members of the committee – Azalee Bostroem, Rodolfo Montez, and Phil Rosenfield – and from Erin Becker, Brett Morris and Adrian Price-Whelan.

The original format of this material is a series of Jupyter notebooks. Using the links below, you can read the notebooks on NBViewer or run them on Colab. If you want to run the notebooks in your own environment, you can download them from this repository and follow the instructions below to set up your environment.

This material is also available in the form of Carpentries lessons, but you should be aware that these versions might diverge in the future.

Prerequisites

This material should be accessible to people familiar with basic Python, but not necessarily the libraries we will use, like Astropy or Pandas. If you are familiar with Python lists and dictionaries, and you know how to write a function that takes parameters and returns a value, that should be enough.

We assume that you are familiar with astronomy at the undergraduate level, but we will not assume specialized knowledge of the datasets or analysis methods we'll use.

Notebook 1

This notebook demonstrates the following steps:

1. Making a connection to the Gaia server,
2. Exploring information about the database and the tables it contains,
3. Writing a query and sending it to the server, and finally
4. Downloading the response from the server as an Astropy `Table`.

Run Notebook 1 on Colab

or click here to read it on NBViewer

Notebook 2

This notebook starts with an example that does a "cone search"; that is, it selects stars that appear in a circular region of the sky.

Then, to select stars in the vicinity of GD-1, we:

- Use `Quantity` objects to represent measurements with units.
- Use the `Gala` library to convert coordinates from one frame to another.
- Use the ADQL keywords `POLYGON`, `CONTAINS`, and `POINT` to select stars that fall within a polygonal region.
- Submit a query and download the results.
- Store the results in a FITS file.

[Run Notebook 2 on Colab]

[or click here to read it on NBViewer]

Notebook 3

Here are the steps in this notebook:

1. We'll read back the results from the previous notebook, which we saved in a FITS file.
2. Then we'll transform the coordinates and proper motion data from ICRS back to the coordinate frame of GD-1.
3. We'll put those results into a Pandas `DataFrame`, which we'll use to select stars near the centerline of GD-1.
4. Plotting the proper motion of those stars, we'll identify a region of proper motion for stars that are likely to be in GD-1.
5. Finally, we'll select and plot the stars whose proper motion is in that region.

[Run Notebook 3 on Colab]

[or click here to read it on NBViewer]

Notebook 4

Here are the steps in this notebook:

1. Using data from the previous notebook, we'll identify the values of proper motion for stars likely to be in GD-1.
2. Then we'll compose an ADQL query that selects stars based on proper motion, so we can download only the data we need.
3. We'll also see how to write the results to a CSV file.

That will make it possible to search a bigger region of the sky in a single query.

[Run Notebook 4 on Colab]

[or click here to read it on NBViewer]

Notebook 5

Here are the steps in this notebook:

1. We'll reload the candidate stars we identified in the previous notebook.
2. Then we'll run a query on the Gaia server that uploads the table of candidates and uses a `JOIN` operation to select photometry data for the candidate stars.
3. We'll write the results to a file for use in the next notebook.

[Run Notebook 5 on Colab]

[or click here to read it on NBViewer]

Notebook 6

Here are the steps in this notebook:

1. We'll reload the data from the previous notebook and make a color-magnitude diagram.
2. Then we'll specify a polygon in the diagram that contains stars with the photometry we expect.
3. Then we'll merge the photometry data with the list of candidate stars, storing the result in a Pandas `DataFrame`.

[Run Notebook 6 on Colab]

[or click here to read it on NBViewer]

Notebook 7

Here are the steps in this notebook:

1. Starting with the figure from the previous notebook, we'll add annotations to present the results more clearly.
2. The we'll see several ways to customize figures to make them more appealing and effective.

3. Finally, we'll see how to make a figure with multiple panels or subplots.

Run Notebook 7 on Colab

or click here to read it on NBViewer

Installation instructions

If you plan to run these notebooks on Colab, you don't have to install anything; you can use the links in the previous section to open and run them. If you want to run the notebooks in your own environment, you might have to do some setup.

You will need to install Python, Jupyter, and some additional libraries. If you don't already have Jupyter, we recommend installing Anaconda, which is a Python distribution that contains everything you need to run the workshop code.
It is easy to install on Windows, Mac, and Linux, and because it does a user-level install, it will not interfere with other Python installations.

Information about installing Anaconda is here.

If you have the choice of Python 2 or 3, choose Python 3.

Now, there are two ways to get the libraries you need:

- Option 1: You can install them in an existing Conda environment.
- Option 2: You can create a new Conda environment.

Installing libraries in an existing environment is simpler, but if you use the same environment for many projects, it will get big, complicated, and prone to package conflicts.

Option 1: *Installing libraries in an existing Conda environment*

Most of the libraries we need can be installed using Conda, by running the following commands in a Terminal. If you are on a Mac or Linux machine, you should be able to use any Terminal.
If you are on Windows, you might have to use the Anaconda Prompt, which you can find under the Start menu.

```
conda install jupyter numpy scipy pandas pytables matplotlib seaborn libopenblas
conda install -c conda-forge astropy astroquery gala python-wget
```

Option 2: *Creating a new Conda environment*

To create a new Conda environment, you'll need to download an environment file from our repository. On Mac or Linux, you can download it using `wget` on the command line:

```
wget https://raw.githubusercontent.com/AllenDowney/AstronomicalData/main/environment.yml
```

Or you can download it using this link.

In a Terminal or Jupyter Prompt, make sure you are in folder where `environment.yml` is stored, and run:

```
conda env create -f environment.yml
```

Then, to activate the environment you just created, run:

```
conda activate AstronomicalData
```

Run Jupyter

If you are not familiar with Jupyter, you can run a tutorial by clicking here. Then select "Try Classic Notebook". It will open a notebook with instructions for getting started. Or you can run this introductory notebook on Colab.

Before you launch Jupyter, download this notebook, which contains code to test your environment.

Or you can use `wget` to download it on the command line, like this:

```
wget https://raw.githubusercontent.com/AllenDowney/AstronomicalData/main/test_setup.ipynb
```

To start Jupyter, run:

```
jupyter notebook
```

Jupyter should launch your default browser or open a tab in an existing browser window. If not, the Jupyter server should print a URL you can use. For example, when I launch Jupyter, I get

```
$ jupyter notebook
[I 10:03:20.115 NotebookApp] Serving notebooks from local directory: /home/username
[I 10:03:20.115 NotebookApp] 0 active kernels
[I 10:03:20.115 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/
[I 10:03:20.115 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to
skip confirmation).
```

In this example, the URL is http://localhost:8888.
When you start your server, you might get a different URL. Whatever it is, if you paste it into a browser, you should should see a home page with a list of directories.

Now open the notebook you downloaded and run the cells that contain `import` statements. If they work and you get no error messages, **you are all set**.

If you get error messages about missing packages, you can install the packages you need using Conda or `pip`.

If you run into problems with these instructions, let us know and we will make corrections. Good luck!

By Allen B. Downey