HW 10 - Astropy

Due April 15th, 2026 at 11:59 PM

Overview

In this homework assignment, you will practice cleaning data before analyzing it with `astropy`. You will also process and analyze FITS (Flexible Image Transport System) files.

Note: FITS files are a common format for astronomical images.

1 Branching with GitHub

Before starting this assignment, create a new Git branch and folder for this assignment:

Steps:

1. Create a branch called `homework10` in your `yourname` repository.

2. Inside that branch, create a folder named `homework10`.

3. Take a screenshot showing that you:
   - Created that branch
   - Deleted all the current files/folders
   - Committed and push your changes to GitHub

4. Save the screenshot as `hw10_branch` inside your `homework10` folder.

2 Units with Astropy

In this problem, we will be using the `solar_system.csv` file from Homework 9. You can find it in the `homework9/` folder on the `main` branch of the `course_assignments` repository.

Before using `astropy` units, we need to "clean" the data.

2.1 Clean the Data

To "clean" this dataset, we will fix formatting issues and convert string values to actual numbers. The term **clean** is being used very loosely here. In general, cleaning data involves removing irrelevant, inconsistent or incorrect information.

2.1.1 Fix Formatting

To start, we need to **transpose**, of the DataFrame so that each row, instead of a parameter, represents a planet.

Steps:

1. Create a new Jupyter Notebook in your `homework10/` folder. Name it `homework10.ipynb`.

2. Import the following libraries:
   - `numpy as np`
   - `pandas as pd`
   - `matplotlib.pyplot as plt`
   - `from astropy import units as u`
   - `from astropy import constants as const`

3. Load the `solar_system.csv` file as a `pandas` DataFrame.

4. Transpose the data so that each row represents a planet:

```
df = df.set_index("Attribute").T
df.index.name = "Planet"
df.reset_index(inplace=True)
df.columns.name = None
```

5. Answer the following:
   (a) Print the shape of the DataFrame **before and after** transposing.
   (b) How many rows/columns are there **before and after** transposing?
   (c) Explain what each line of code above does.
   Hint: Print the shape of the DataFrame after each line of code.
   (d) Why isn't the new shape `(11, 20)`? What's different about the structure?
   (e) Print the new column names.
   (f) How many columns have units? List them. How many don't? List them.
   (g) Print the whole **transposed** DataFrame.

2.1.2 Investigate Data Types

Even though your table looks like it has numbers, many columns are actually stored as strings.

Steps:

1. Use `df.dtypes` to check the current data types of **entire columns** with `pandas`. You will see **all** of the columns are `object`, which means they could be strings, lists or a mix of data types.

2. Loop through the columns to inspect the individual data types of the values:
   Hint:

   ```
   df[col].apply(type)
   ```

3. Answer the following:
   (a) What types are the actual values?
   (b) Why is storing numeric values as strings a problem for data analysis?

2.1.3 Convert Strings to Numbers

Steps:

1. For each column (except `"Planet"`, `"Ring System?"` and `"Global Magnetic Field?"`), use:

   ```
   pd.to_numeric(df[col], errors="coerce")
   ```

2. Check the data types again with `df.dtypes`.

3. Answer the following:
   (a) What are the **three** different data types now in the DataFrame?
   (b) Define each one (e.g., `float64`).
   (c) Which columns are still `object`, and why?

2.2 Apply Astropy Units

Now that your data is 'cleaned', we'll use Astropy to attach physical units.

Steps:

1. Define a function called `attach_units()` that:
   - Takes a column name
   - Applies an Astropy unit
   - (If applicable) renames the column using `df.rename()`

Example of how to apply units to one column:

```
unit = 1e24 * u.kg
old_col = "Mass (10^24kg)"
new_col = "Mass (kg)"

new_values = []
for val in df[old_col]:
    new_values.append(val)

df[old_col] = new_values
df.rename(columns={old_col: new_col}, inplace=True)
```

2. Pick your favorite column and apply your function. Print the column afterward.
   Hint: Your values should now include units like: `3.3e+23 kg`.

2.3 Data Analysis

Now you will practice integrating Astropy units with your `pandas` DataFrame. Perform the tasks below:

1. **Semi-Major Axis:**
   - Like Homework 9, compute the average of **perihelion** and **aphelion** to get the semi-major axis.
   - Attach this as a new column right after `Aphelion` column using `df.insert()`.
   - To confirm the column is inserted, print all the column names.
   - Print the new column and confirm it has units.

2. **Convert Orbital Period to Years:**
   - Convert orbital period from days to years with `astropy`.
   - Change the units in the column name.
   - Print all the column names.
   - Print the full column.
   - Print your favorite planet's orbital period in years, formatted to **four** decimal places using an f-string.

3. **Convert Distance Columns (km $\longrightarrow$ AU):**
   - Define **AU** in your own words.
   - Print **one AU** with Astropy: `const.au`.
   - How many kilometers are in one AU? Use Astropy to answer for you.
   - Convert all distance columns in kilometers to AU.

- Rename the columns (e.g., `"Diameter (km)"` becomes `"Diameter (AU)"`)
- Print a sample value from each converted column for your favorite planet.

4. **Bonus! (optional but *highly* recommended)**
   - Plot surface gravity (meters per second squared) vs. mass (in grams) using Matplotlib:

$$\text{Surface Gravity } \left[\frac{\text{m}}{\text{s}^2}\right] = \frac{GM}{R^2}$$

   where G is the gravitational constant, M is the mass, and R is the radius in meters.
   Hint: Plot axes on a logarithmic scale for a better representation.

5. **Bonus Bonus!! (optional but *highly* recommended)**
   - Use `scipy.optimize.curve_fit` to find a line of best fit in the plot above.
     Hint: If you plot with a logarithmic scale, you can perform a linear fit.
   - Print the best fit parameters.
   - Print the reduced chi-squared.
   - Plot the scattered data with the fit.
   - Do you have a good fit?

2.4 Save Your Results

Steps:

1. Save the new DataFrame (with all the changes) as `units.csv`:

```
df.to_csv("units.csv", index=False)
```

2. Save this file in your `homework10/` folder in your `homework10` branch.

3 FITS Files

Pull the latest `main` branch of the `course_assignment` repository. Inside the `homework10/` folder, you will find **five** FITS files:

- `M13_00001.fits`
- `M13_00002.fits`
- `M13_00003.fits`
- `M13_00004.fits`
- `M13_00005.fits`

These are astronomical images. We will work on displaying them and combining them.

3.1 Read FITS Files

Steps:

1. Create a new Jupyter Notebook in your `homework10/` folder. Name it `fits.ipynb`.

2. Import the following libraries:
   - `matplotlib.pyplot as plt`
   - `from astropy.io import fits`
   - `numpy as np`

3. Write a loop that uses `astropy.io.fits` to load the all **five** FITS files. Here's an example for loading a single file:

```
with fits.open("path/to/M13_0001.fits") as hdul:
    header = hdul[0].header
    data = hdul[0].data
```

   Hint: Use a `for` loop and append all the data to a list.

3.2 Plot Images

When we plot the FITS files in their current state, it will be pretty difficult to see the star.

Steps:

1. Make a **1x5 subplots** using `plt.imshow()` to plot each of the FITS files:
   - Use `origin="lower"`

2. Make a second set of **1x5 subplots**, but this time:
   - Use the same color scale across all plots (`vmin`, `vmax`)
   - Show the color bar
   - What differences do you notice in the images compared to the previous subplots?

3.3 Plot Histograms

Steps:

1. For each FITS file, make a histogram of pixel values:

```
plt.hist(data.ravel(), bins=50)
```

2. Answer the following:

(a) What value is on the x-axis? y-axis?

(b) What does `bins=50` mean?

(c) Where is most of the data centered for pixel intensity?

(d) How spread out is the data? Ignore the outliers.

(e) What is the median/mean pixel intensity value for each image?

3.4 Combine and Normalize Images

Astronomers combine and stack images to improve the quality of their data. It helps reveal fainter details and average out random noise.

Steps:

1. Combine your images by taking the **median across axis 0** using `np.median()`:
   - What's the new median pixel intensity value of your combined image?

2. Plot a histogram of the combined file.

3. Use the histogram's start and end values as `vmin` and `vmax` in your `plt.imshow()`. Don't use the entire axis for your `vmin` and `vmax` values.
   Hint: The difference between your `vmin` and `vmax` should be around 100.

4. How dose this image compare to the earlier versions?

5. Save the combined image as `combined.fits` using:

```
fits.writeto("combined.fits", combined_image, overwrite=True)
```

3.5 Bonus

Steps:

1. Replot the **original five** FITS images again but use the `vmin` and `vmax` values from the combined histogram.

4 Submitting Your Homework

Make sure before you submit your repository that you have fully run your Jupyter Notebooks, we want to be able to see the output when we are grading!

Steps:

1. In your `yourname/` repository, run:

```
git add .
git commit -m "done with hw10"
git push origin homework10
```

2. Take a screenshot of the terminal output.
3. Name the screenshot: `hw10_changes`.
4. Place it inside your `homework10/` folder.

5. Ensure all screenshots are saved correctly and your code runs without errors. Your `homework10/` folder should now contain:

```
homework10/
|--- units.csv
|--- combined.fits
|--- homework10.ipynb
|--- hw10_branch.png
|--- hw10_changes.png
```

6. Go to Gradescope and find the **Homework 10: Astropy** assignment.
7. Select the option to upload a GitHub repository.
8. Make sure to upload your `homework10` branch.
9. Submit your `yourname` repository.

Great job!