HW 7 - Plotting

Due March 18th, 2026 at 11:59pm

Overview

In this homework assignment, you'll practice plotting with Matplotlib and importing/analyzing data with NumPy.

1 Branching with GitHub

Just like the last homework, create a branch called `homework7` in your `yourname` repository. Inside that branch, create a folder called `homework7`. Please work on this homework assignment on that branch and in this folder.

Take a screenshot of making this new branch, removing all of the files and adding/committing/pushing it to GitHub.

Name the screenshot: `hw7_branch`. Save it inside your `homework7/` folder in your `homework7/` branch.

2 Matplotlib

Start by opening a Jupyter Notebook and naming it `homework7.ipynb`. At the top of your notebook, call:

```
import numpy as np
import matplotlib.pyplot as plt
```

Then complete the following:

1. Write a function called `make_sine_wave(x, A, w)` that returns: $y = A \cdot sin(\omega x)$. Where x is an array of input values, A is the amplitude, and ω is the angular frequency.

2. Use `np.linspace()` to create an array that goes from 0 to 2π with 1000 points, call it `x`.

3. Create two arrays:
   - `amplitude = [0.5, 1, 1.5, 2, 2.5]`

- `frequency = [1, 2, 3, 4, 5]`

4. Create three side-by-side subplots `figsize=(25, 5)`.

5. Loop through both arrays simultaneously with the built-in `zip()` function.
6. Call your `make_sine_wave()` function for each subplot.
   - First subplot: Vary only the amplitude
   - Second subplot: Vary only the frequency
   - Third subplot: Vary the amplitude and frequency together.

   Hint: When you aren't varying a variable, just set it to 1.

7. Add descriptive titles, axis labels and a legend to each subplot.
   Hint: I recommend increasing the font size of your labels.
8. Call `plt.show()` to display the plots.

Bonus:

- For each line, plot it with a different linestyle.
- Add a legend to each plot that displays the frequency and amplitude.

3 Plot a Hertzsprung-Russell Diagram!

To access the data for this problem, pull the latest version of the `main` branch of the `course_assignments` repository. Typically, you were asked to copy files over, but no need for this assignment. Just write a relative path to point to it in your Jupyter Notebook. More information below.

For context, next homework you will start working with a package called `pandas` which makes importing, reducing and analyzing data much more convenient. But it is still very much possible to do that with `numpy`.

3.1 Importing Data with NumPy

Steps:

1. Pull the latest version of the `main` branch of the `course_assignments` repository. Call:

   ```
   git pull origin main
   ```

   You will see a folder called `homework7` containing a file called `star_data.txt`.

2. In your `yourname` repository, under your `homework7` branch, in your `homework7` folder, create a new Jupyter Notebook called: `hr_diagram.ipynb`

3. Import `numpy` and `matplotlib` at the top of your `hr_diagram.ipynb` Notebook.

4. Use `np.loadtxt()` to load the `.txt` file. For example:

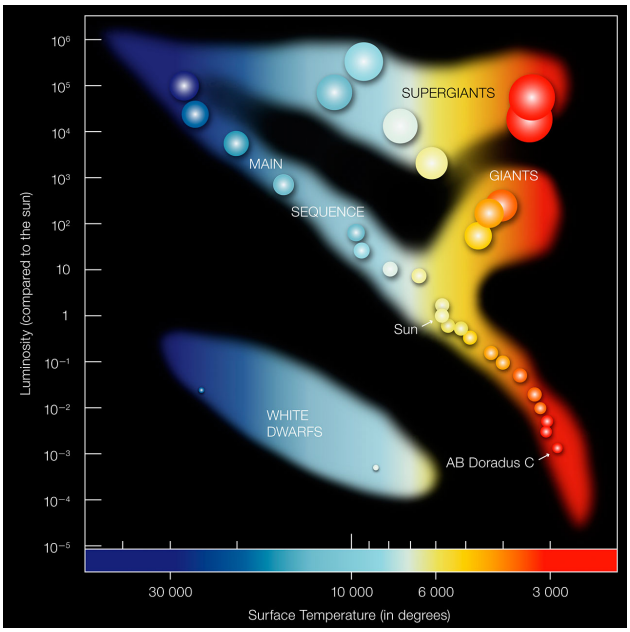
```
data = np.loadtxt("../../course_assignments/homework6/star_data.txt",
                  skiprows=1)
```

   Your file path might vary slightly if you are not following the structure from class. **Make sure to include the argument** `skiprows=1`.

Note: Using `np.loadtxt()` automatically loads your data as a NumPy array.

3.2 Analyzing Imported Data

You just imported a file called `star_data.txt` into your Jupyter Notebook. This file contains data on the magnitude and temperature of different stars. With this data, you will make an Hertzsprung-Russell (HR) Diagram. Here is an example:



Steps:

1. Print the shape of the data. How many rows are there?

2. The dataset comes with two columns:
   - **1st column:** the magnitude (brightness) of the star in units of magnitude.
   - **2nd column:** the temperature of the star in units of Kelvin.

   Note: Each row in this dataset is a different star.

3. Create two variables: `magnitude` and `temperature`
    - Set `magnitude` equal to the first column.
    - Set `temperature` equal to the second column.

    Hint: Use array slicing.

4. Print the **minimum** and **maximum** value of the `magnitude` variable. What stars do they correspond to?
5. Print the **minimum** and **maximum** value of the `temperature` variable. What stars do they correspond to?
6. Recall: magnitude is backwards. Larger numbers correspond to dimmer stars. Which star is brightest? Which star is the dimmest?

3.3 Plot and HR Diagram

Some notes about HR diagrams:

- The x-axis and y-axis are on a **log** scale, not linear.
- **Temperature** is plotted backwards. With hotter temperatures towards the origin.
- **Magnitude** is plotted backwards. With larger numbers toward the origin. Practically, this is because magnitude in-of-itself is backwards.

Overall, in astronomy, we plot stuff weird because of historical reasons.

Steps:

1. Create two new variables:
    - `log_magnitude = np.log(magnitude)`
    - `log_temperature = np.log(temperature)`
2. Create a single figure with a figure size of `figsize=(8, 6)`.
3. **Scatter** the `log_magnitude` variable on the y-axis and the `log_temperature` on the x-axis.
4. Invert the x- and y-axis.
   Hint: After scattering the points, call: `plt.gca().invert_xaxis()` *and* `plt.gca().invert_yaxis()`
5. Highlight the stars you found above by plotting them again with different colors and markers:
    - For the brightest star, make is a black star.
    - For the dimmest star, make is a purple star.
    - For the hottest star, make is green star.
    - For the coldest star, make is orange star.

Hint: You can change the color by calling `color="black"`*, you can change the marker by calling* `marker="*"` *and you can also change the size by calling* `s = 300`*.* ***Don't forget to plot them on a log scale!***

6. Add a legend. Includes a label for stars in general and one each for the brightest/dimmest/hottest and coldest.

7. Add a title: "Hertzsprung-Russel Diagram".

8. Add an x-label: "Temperature (log K)"

9. Add an y-label: "Magnitude".

Bonus:

Add a colormap that shows goes from coldest stars to hottest stars.

4 Notice

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!

5 Submitting Your Homework

After completing everything:

Steps:

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

```
git add .
git commit -m "done with hw7"
git push origin homework7
```

2. Take a screenshot of the terminal output.
3. Name the screenshot: `hw7_changes`.
4. Place it inside your `homework7/` folder.

5. Ensure all screenshots are saved correctly and your code runs without errors. Your `homework7/` folder should now contain:

```
homework7/
|--- homework7.ipynb
|--- hr_diagram.ipynb
|--- hw7_branch.png
```

```
|--- hw7_changes.png
```

6. Go to Gradescope and find the **Homework 7: Plotting** assignment.
7. Select the option to upload a GitHub repository.
8. Make sure to upload your `homework7` branch.
9. Submit your `yourname` repository.

Great job!