Python, Machine Learning, and Astronomy

Dr. Brian R. Kent, Director of Science Communications
National Radio Astronomy Observatory
Twitter & Instagram: @VizAstro
YouTube: Visualize Astronomy   June 2024

Overview

- History of Python
- Why use Python in Astronomy?
- Why Revision Control is important...
- Google Colab and simple AI examples
- Data Colorimetry and Presentation
- Machine Learning Example with Tensorflow
- Resources to explore further
- Conferences (in-person and remote!)
- **Recommendations**

- High level programming language that is scriptable and supports many programming styles.
- Created in the late 80s/early 90s by *Guido van Rossum*, named after *Monty Python*
- The **Python Standard Library** can be used to accomplish a wide variety of tasks - always check there first!

https://ivastar.github.io/python-novice-astro/

tiobe.com - Usage of Languages 2018

Jun 2018	Jun 2017	Change	Programming Language	Ratings	Change
1	1		Java	15.368%	+0.88%
2	2		C	14.936%	+8.09%
3	3		C++	8.337%	+2.61%
4	4		Python	5.761%	+1.43%
5	5		C#	4.314%	+0.78%
6	6		Visual Basic .NET	3.762%	+0.65%
7	8	⌃	PHP	2.881%	+0.11%
8	7	⌄	JavaScript	2.495%	-0.53%
9	-	⌃⌃	SQL	2.339%	+2.34%
10	14	⌃⌃	R	1.452%	-0.70%
11	11		Ruby	1.253%	-0.97%
12	18	⌃⌃	Objective-C	1.181%	-0.78%
13	16	⌃	Visual Basic	1.154%	-0.86%
14	9	⌄⌄	Perl	1.147%	-1.16%
15	12	⌄	Swift	1.145%	-1.06%
16	10	⌄⌄	Assembly language	0.915%	-1.34%
17	17		MATLAB	0.894%	-1.10%
18	15	⌄	Go	0.879%	-1.17%
19	13	⌄⌄	Delphi/Object Pascal	0.875%	-1.28%
20	20		PL/SQL	0.848%	-0.72%

TIOBE Index for June 2019

June Headline: Python continues to soar in the TIOBE index

This month Python has reached again an all time high in TIOBE index of 8.5%. If Python can keep this pace, it will probably replace C and Java in 3 to 4 years time, thus becoming the most popular programming language of the world. The main reason for this is that software engineering is booming. It attracts lots of newcomers to the field. Java's way of programming is too verbose for beginners. In order to fully understand and run a simple program such as "hello world" in Java you need to have knowledge of classes, static methods and packages. In C this is a bit easier, but then you will be hit in the face with explicit memory management. In Python this is just a one-liner. Enough said.

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Jun 2019	Jun 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.004%	-0.36%
2	2		C	13.300%	-1.64%
3	4	⌃	Python	8.530%	+2.77%
4	3	⌄	C++	7.384%	-0.95%
5	6	⌃	Visual Basic .NET	4.624%	+0.86%
6	5	⌄	C#	4.483%	+0.17%
7	8	⌃	JavaScript	2.716%	+0.22%
8	7	⌄	PHP	2.567%	-0.31%
9	9		SQL	2.224%	-0.12%
10	16	⌃⌃	Assembly language	1.479%	+0.56%
11	15	⌃⌃	Swift	1.419%	+0.27%
12	12		Objective-C	1.391%	+0.21%
13	11	⌄	Ruby	1.388%	+0.13%
14	60	⌃⌃	Groovy	1.300%	+1.11%
15	18	⌃	Go	1.257%	+0.38%
16	14	⌄	Perl	1.173%	+0.03%
17	19	⌃	Delphi/Object Pascal	1.129%	+0.25%
18	17	⌄	MATLAB	1.077%	+0.18%
19	13	⌄⌄	Visual Basic	1.069%	-0.08%
20	20		PL/SQL	0.929%	+0.08%

Jun 2020	Jun 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	17.19%	+3.89%
2	1	▼	Java	16.10%	+1.10%
3	3		Python	8.36%	-0.16%
4	4		C++	5.95%	-1.43%
5	6	▲	C#	4.73%	+0.24%
6	5	▼	Visual Basic	4.69%	+0.07%
7	7		JavaScript	2.27%	-0.44%
8	8		PHP	2.26%	-0.30%
9	22	▲▲	R	2.19%	+1.27%
10	9	▼	SQL	1.73%	-0.50%
11	11		Swift	1.46%	+0.04%
12	15	▲	Go	1.02%	-0.24%
13	13		Ruby	0.98%	-0.41%
14	10	▼▼	Assembly language	0.97%	-0.51%
15	18	▲	MATLAB	0.90%	-0.18%
16	16		Perl	0.82%	-0.36%
17	20	▲	PL/SQL	0.74%	-0.19%
18	26	▲▲	Scratch	0.73%	+0.20%
19	19		Classic Visual Basic	0.65%	-0.42%
20	38	▲▲	Rust	0.64%	+0.38%

TIOBE Index for June 2021

June Headline: Python has never been so close to position #1 before

Python is about to take over the first position in the TIOBE index. The gap between the current number one, programming language C, and Python is only 0.7% now. Next month, the TIOBE index is celebrating its 20-year anniversary. Programming languages C and Java are the only 2 languages that reached a number 1 position during these 20 years. So if Python is going to take over the first position in the TIOBE index, this will certainly be a historical moment, which is worth celebrating. There appear to be hardly any interesting moves further down the chart. Possible future champions such as Dart, Kotlin, Julia, Rust, TypeScript, and Elixir didn't show any significant changes last month. -- *Paul Jansen CEO TIOBE Software*

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Jun 2021	Jun 2020	Change	Programming Language	Ratings	Change
1	1		C	12.54%	-4.65%
2	3	^	Python	11.84%	+3.48%
3	2	v	Java	11.54%	-4.56%
4	4		C++	7.36%	+1.41%
5	5		C#	4.33%	-0.40%
6	6		Visual Basic	4.01%	-0.68%
7	7		JavaScript	2.33%	+0.06%
8	8		PHP	2.21%	-0.05%
9	14	^	Assembly language	2.05%	+1.09%

Tiobe index for June 2022

Jun 2022	Jun 2021	Change	Programming Language	Ratings	Change
1	2	^	Python	12.20%	+0.35%
2	1	v	C	11.91%	−0.64%
3	3		Java	10.47%	−1.07%
4	4		C++	9.63%	+2.26%
5	5		C#	6.12%	+1.79%
6	6		Visual Basic	5.42%	+1.40%
7	7		JavaScript	2.09%	−0.24%
8	10	^	SQL	1.94%	+0.06%
9	9		Assembly language	1.85%	−0.21%
10	16	^	Swift	1.55%	+0.44%

Tiobe index for June 2023

Jun 2023	Jun 2022	Change		Programming Language	Ratings	Change
1	1			Python	12.46%	+0.26%
2	2			C	12.37%	+0.46%
3	4	⌃		C++	11.36%	+1.73%
4	3	⌄		Java	11.28%	+0.81%
5	5			C#	6.71%	+0.59%
6	6			Visual Basic	3.34%	-2.08%
7	7			JavaScript	2.82%	+0.73%
8	13	⌃⌃		PHP	1.74%	+0.49%
9	8	⌄		SQL	1.47%	-0.47%
10	9	⌄		Assembly language	1.29%	-0.56%

Tiobe index for June 2024

Jun 2024	Jun 2023	Change		Programming Language	Ratings	Change
1	1			Python	15.39%	+2.93%
2	3	^		C++	10.03%	-1.33%
3	2	v		C	9.23%	-3.14%
4	4			Java	8.40%	-2.88%
5	5			C#	6.65%	-0.06%
6	7	^		JavaScript	3.32%	+0.51%
7	14	^^		Go	1.93%	+0.93%
8	9	^		SQL	1.75%	+0.28%
9	6	v		Visual Basic	1.66%	-1.67%
10	15	^^		Fortran	1.53%	+0.53%

Python and Astronomy

- With iPython, it is used as the command line shell & interpreter for CASA
- Used for the framework for VLA/GBT imaging/ALMA pipelines
- Can interface with AIPS via Obit
- Can interface with IRAF via PyRAF
- Used in PRESTO - pulsar data reduction
- Many modules and libraries available - numpy, matplotlib, Kapteyn, astropy, AplPy, etc.
- Managed via pip or Anaconda
- Scriptable for astronomers OOP for engineers
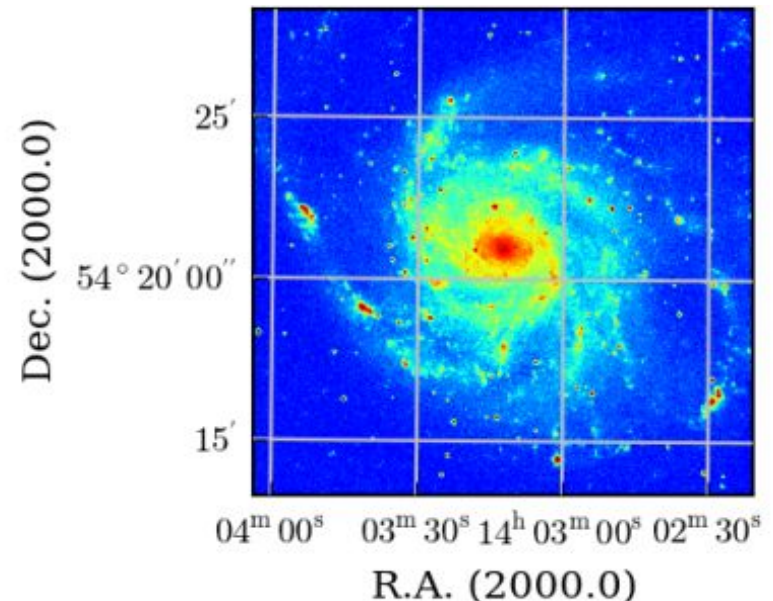
Popular Python Resources

iPython - more user friendly shell

Astropy - great for data import, manipulation, catalog queries

ApIPy - general image/coordinate display utility

Matplotlib - general purpose plotting tool

Scipy - numpy and fitting routines (some overlap with astropy…)

Kapteyn - Good for mapping projections

AstroML - Machine Learning   http://www.astroml.org/index.html

PANDAS - Data Analysis Library http://pandas.pydata.org/

Python Data Science Handbook:

   https://jakevdp.github.io/PythonDataScienceHandbook/

Recommendation: **Learn to use these inside and out!**

Where to start?

- https://www.youtube.com/@PyConUS/videos
- Pycon: https://us.pycon.org/2024/ (and video tutorials)
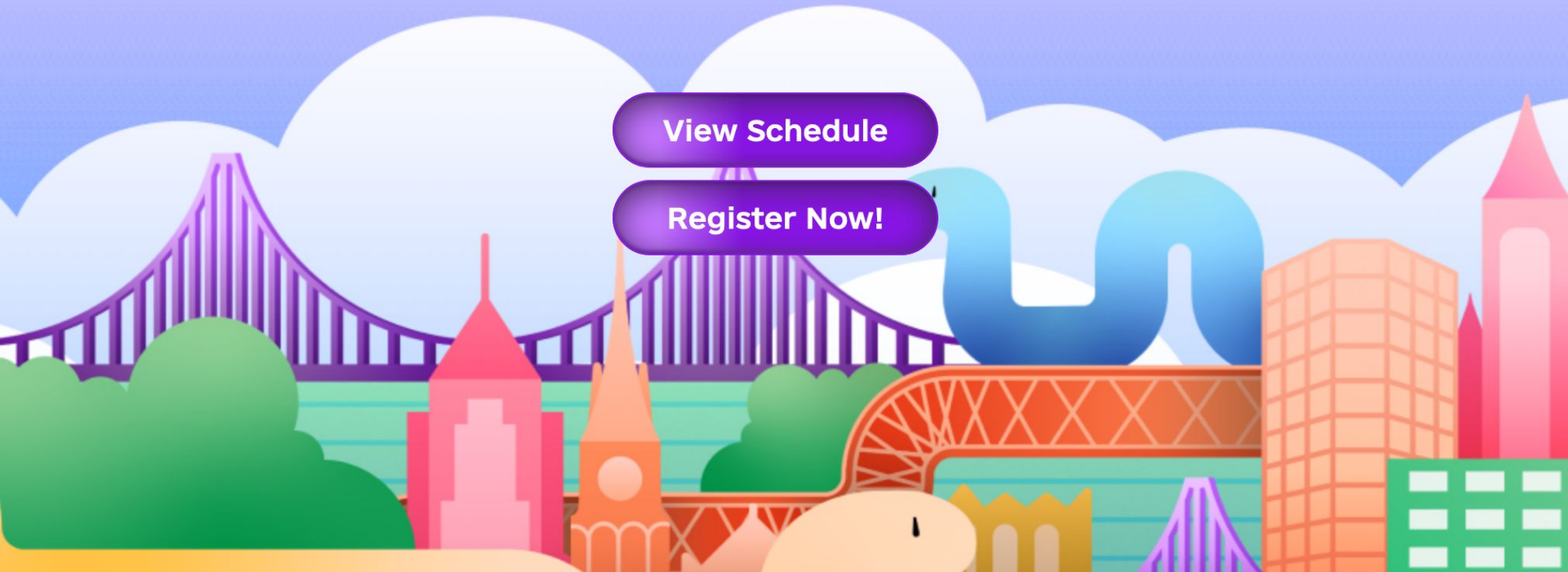
Revision Control

Revision Control for Software

Popular revision control systems (RCS) include SVN, CVS, Mercurial, and **git**.

Tracks changes made to files - good for collaboration among teams or to see when and where changes were made to your code!

Repo for this presentation:

https://github.com/brkent/SummerStudent2024

1. Plotting
2. 2D Plotting
3. Cosmology
4. Fitting Function
5. FITS images
6. VLASS and subimages

git revision control

mkdir sumstudent

cd sumstudent

git clone https://github.com/brkent/SummerStudent2024.git

This will create and link your directory
to the git repository

git revision control

```
#Create a readme file
touch README.md
git add README.md
git commit -m "Initial README commit"
git push -u origin main
```

This is your modern scientific lab notebook!

Recommendation:

If you do not already know how to use git, please take some time this summer and practice with these tutorials:

https://docs.github.com/en/get-started

Google Colab

Colaboratory is a **Google** research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

https://github.com/brkent/SummerStudent2024/blob/main/example04_gaussian_fit_seaborn.ipynb

Also works in github.dev…

Recommendation: Learn to prototype bit of code in the cloud and make **clean** Jupyter notebooks. Make your work portable and shareable! If you can point potential employers at your work and it is organized/documented - all the better!

Example: Astropy Cosmology Calc

https://github.com/brkent/SummerStudent2024/blob/main/example03_cosmology.ipynb
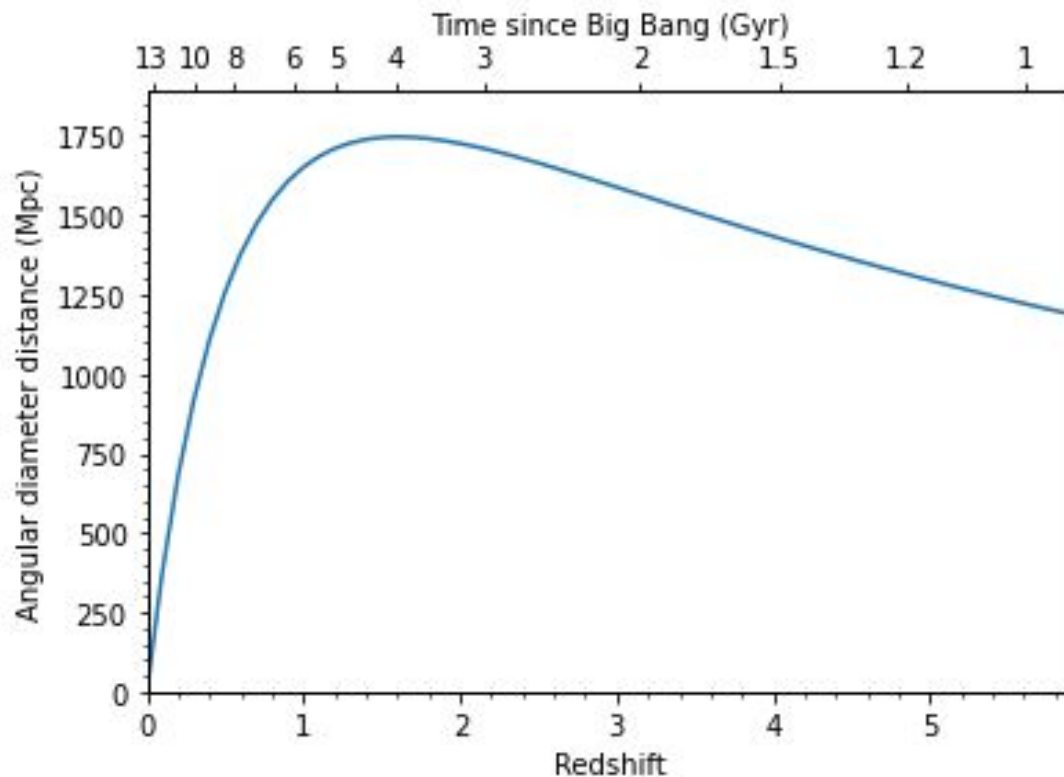
Example: Create a noisy spectrum and fit a Gaussian

https://github.com/brkent/SummerStudent2024/blob/main/example04_gaussian_fit_seaborn.ipynb

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]$$
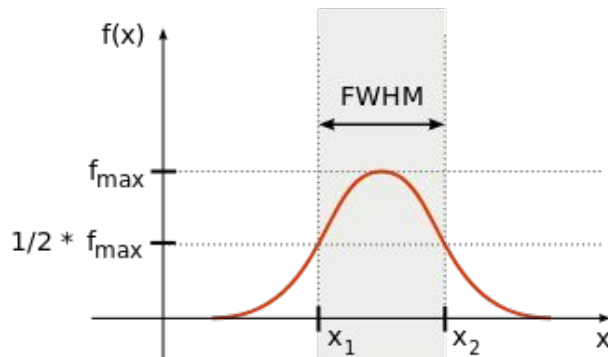
$$\text{FWHM} = 2\sqrt{2\ln 2}\,\sigma \approx 2.355\,\sigma.$$
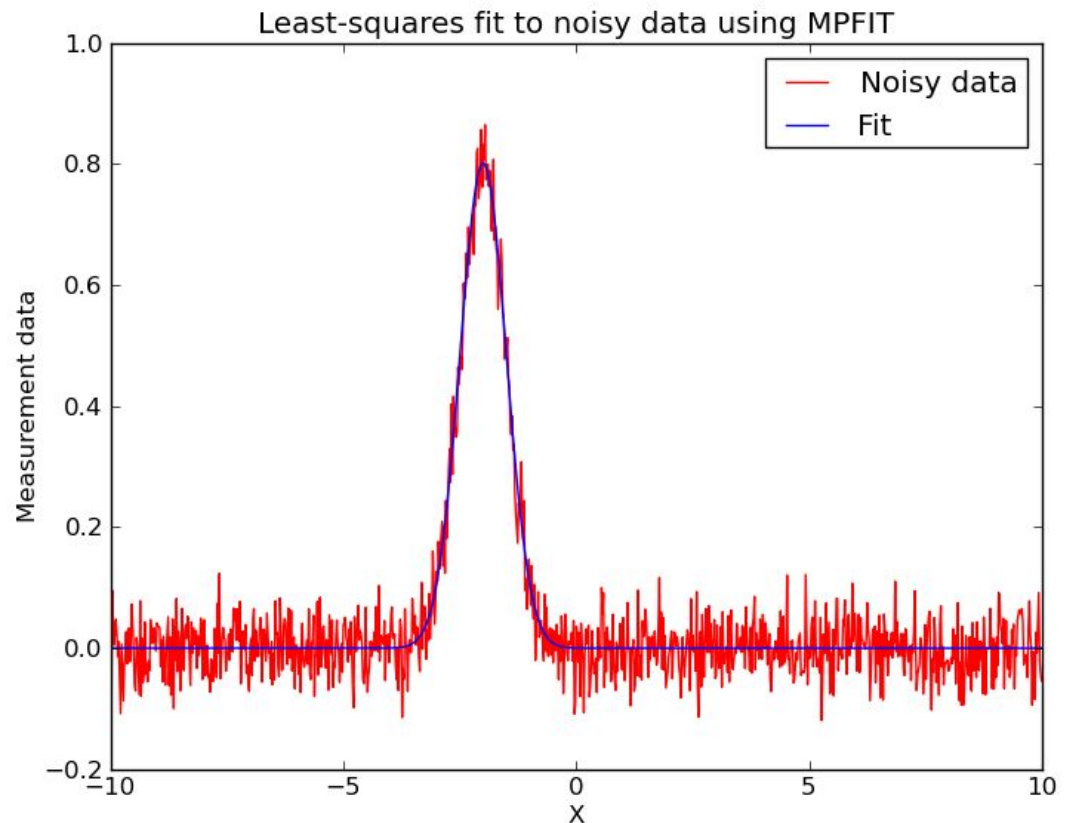
Another series of great fitting examples...

https://learn.astropy.org/tutorials/Models-Quick-Fit.html

What if we had a tool to prototype and make our workflow a bit faster?

BR

Write a Python function that will fit a gaussian to a set of data points and give me the matplotlib plotting code

Certainly! Here's a Python function that fits a Gaussian curve to a set of data points using the SciPy library and provides the corresponding matplotlib plotting code:

```python
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

def fit_gaussian(x, y):
    # Define the Gaussian function
    def gaussian(x, amplitude, mean, stddev):
        return amplitude * np.exp(-((x - mean) / stddev)**2 / 2)

    # Perform the curve fit
    initial_guess = [np.max(y), np.mean(x), np.std(x)]  # Initial guess for
    optimized_params, _ = curve_fit(gaussian, x, y, p0=initial_guess)

    # Generate data points from the fitted Gaussian function
    x_fit = np.linspace(np.min(x), np.max(x), 100)
    y_fit = gaussian(x_fit, *optimized_params)

    # Plot the original data points and the fitted Gaussian curve
    plt.plot(x, y, 'bo', label='Data')
    plt.plot(x_fit, y_fit, 'r-', label='Gaussian Fit')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.show()
```
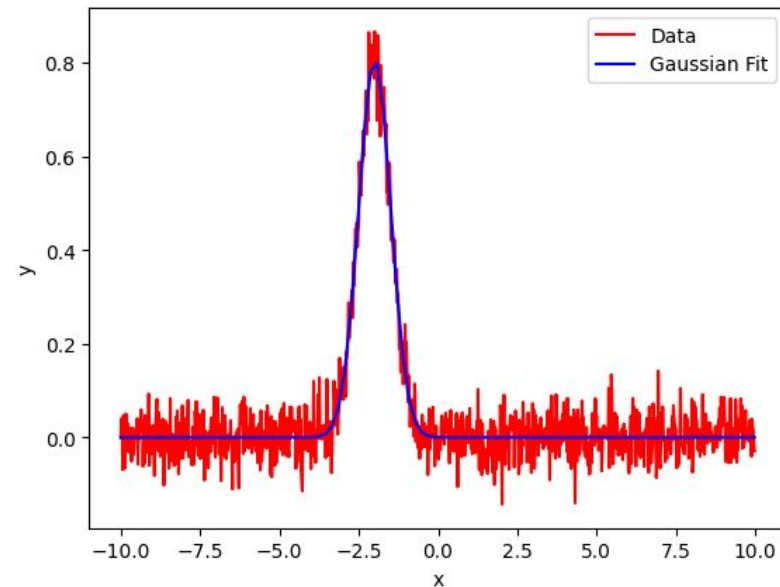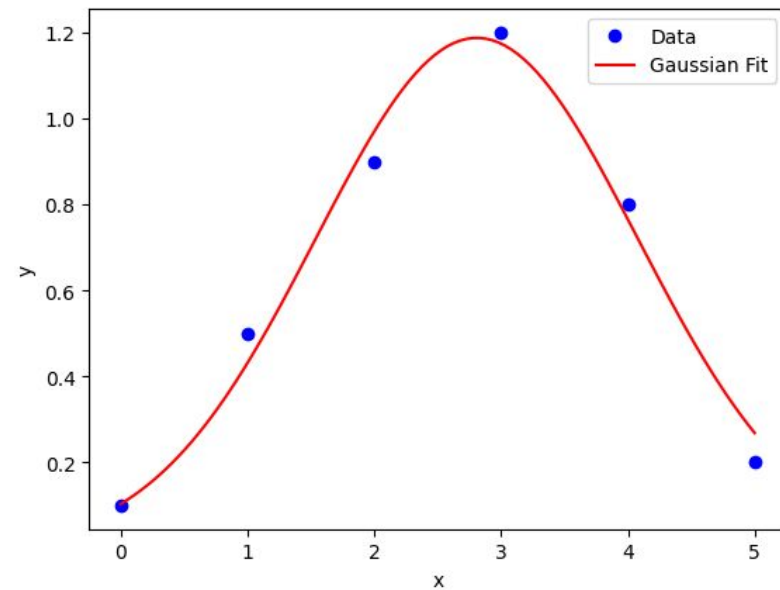
You can use this function by passing your data points as `x` and `y` arguments. Here's an example usage:

```python
x = np.array([0, 1, 2, 3, 4, 5])
y = np.array([0.1, 0.5, 0.9, 1.2, 0.8, 0.2])

fit_gaussian(x, y)
```

Make a cosmology plot in Python of the Angular diameter distance in units of Megaparsecs as a function of redshift using AstroPy

To create a cosmology plot of the Angular Diameter Distance as a function of redshift using AstroPy, you can follow the steps below:

1. Install the necessary libraries:

```
pip install astropy matplotlib
```

2. Import the required modules:

```python
import numpy as np
import matplotlib.pyplot as plt
from astropy.cosmology import Planck18 as cosmo
```

3. Define the redshift range:

```python
redshifts = np.linspace(0, 10, 100)
```

4. Calculate the Angular Diameter Distance for each redshift using the chosen cosmology:

```python
angular_diameter_distances = cosmo.angular_diameter_distance(redshifts)
```

5. Plot the Angular Diameter Distance as a function of redshift:

```python
plt.plot(redshifts, angular_diameter_distances / 1e6)  # Dividing by 1e6
plt.xlabel('Redshift')
plt.ylabel('Angular Diameter Distance (Mpc)')
plt.title('Angular Diameter Distance vs Redshift')
plt.grid(True)
plt.show()
```

Putting it all together, here's the complete code:

```python
import numpy as np
import matplotlib.pyplot as plt
from astropy.cosmology import Planck18 as cosmo

redshifts = np.linspace(0, 10, 100)
angular_diameter_distances = cosmo.angular_diameter_distance(redshifts)

plt.plot(redshifts, angular_diameter_distances / 1e6)  # Dividing by 1e6 to
plt.xlabel('Redshift')
plt.ylabel('Angular Diameter Distance (Mpc)')
plt.title('Angular Diameter Distance vs Redshift')
plt.grid(True)
plt.show()
```

This code will generate a plot showing the Angular Diameter Distance in units of Megaparsecs as a function of redshift, using the Planck 2018 cosmology. Feel free to adjust the redshift range or use a different cosmology as needed.
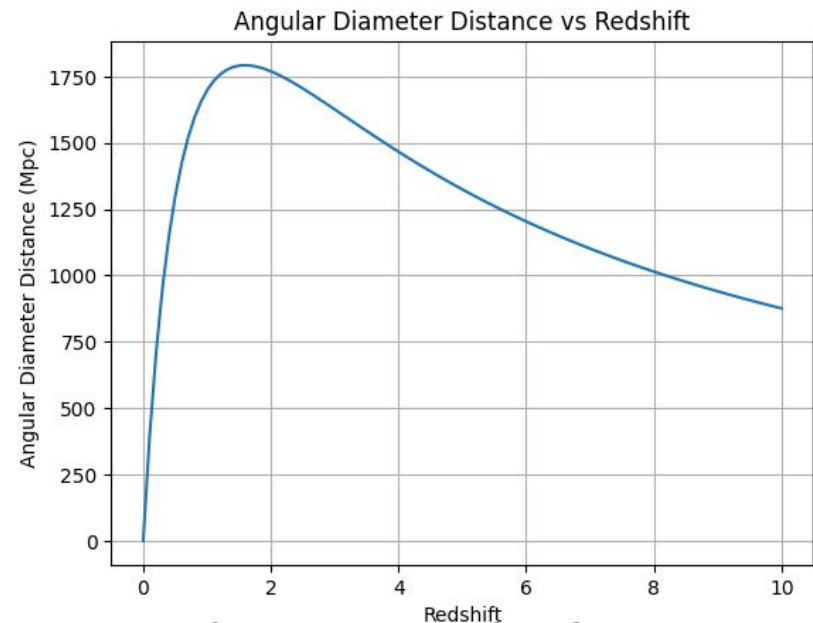
*Had to fix the units with ChatGPT…but it was close!

```python
import matplotlib.pyplot as plt
from astropy.cosmology import WMAP9 as cosmo  # Use WMAP9 cosmology by
import numpy as np

# Define redshift range
z_vals = np.linspace(0, 2, 100)  # Redshift values from 0 to 2 with 100

# Calculate angular diameter distance
ang_diameter_dist = cosmo.angular_diameter_distance(z_vals)

# Plot the data
plt.figure(figsize=(8, 6))
plt.plot(z_vals, ang_diameter_dist.to(u.Mpc), label="Angular Diameter Di

# Set labels and title
plt.xlabel("Redshift (z)")
plt.ylabel("Angular Diameter Distance (Mpc)")
plt.title("Angular Diameter Distance vs. Redshift (WMAP9 Cosmology)")

# Add legend
plt.legend()

# Show the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```

Use code with caution.

This code does the following:

1. Imports necessary libraries: `matplotlib.pyplot` for plotting,
   `astropy.cosmology` for cosmological calculations, and `numpy` for numerical
   operations.
2. Defines a `WMAP9` cosmology object from `astropy.cosmology`. You can change
   this to another pre-defined cosmology or create a custom one.
3. Creates a list of redshift values (`z_vals`) ranging from 0 to 2 with 100 points.
4. Calculates the angular diameter distance for each redshift using
   `cosmo.angular_diameter_distance` and converts the units to Megaparsecs
   (Mpc) using `astropy.units`.
5. Creates a plot using `plt.plot`.
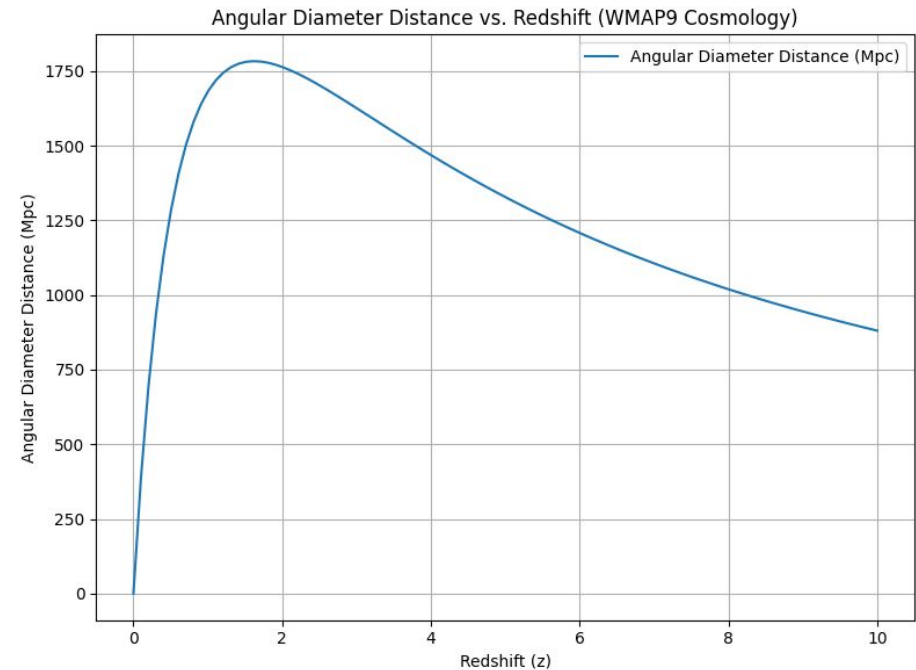
*Google Gemini was close - did not import the units from AstroPy, and only went to a redshift of 2.

BR Write a Python function to download and display an optical image of the galaxy M51

To download and display an optical image of the galaxy M51, you can use the `astroquery` package to retrieve the image from an online database. Here's a Python function that accomplishes this:

```python
import matplotlib.pyplot as plt
from astropy.io import fits
from astroquery.skyview import SkyView

def download_and_display_m51_image():
    # Set the coordinates and image size for M51
    target_name = "M51"
    ra = 202.4697  # Right Ascension in degrees
    dec = 47.1953  # Declination in degrees
    image_size = 20  # Size of the image in arcminutes

    # Query the online database and download the optical image
    images = SkyView.get_images(position=f"{ra} {dec}", survey='DSS', coordi
    image = images[0]

    # Load the downloaded FITS file
    hdulist = fits.open(image[0])
    image_data = hdulist[0].data

    # Display the image
    plt.figure(figsize=(8, 8))
    plt.imshow(image_data, cmap='gray')
    plt.title(target_name)
    plt.axis('off')
    plt.show()
```

To use this function, you need to have the `astroquery` package installed. You can install it via `pip`:

```
pip install astroquery
```

Once installed, you can call the `download_and_display_m51_image()` function to download and display the optical image of the galaxy M51. It will retrieve the image from the Digital Sky Survey (DSS) using the specified coordinates and image size. The resulting image will be displayed using Matplotlib.

```python
download_and_display_m51_image()
```

This will open a new window displaying the optical image of the galaxy M51. Adjust the target coordinates and image size as needed to retrieve images of other celestial objects.

There are some issues with this one - but they are fixable…

Query for an image…

```
!pip install astroquery


import matplotlib.pyplot as plt
from astropy.io import fits
from astroquery.skyview import SkyView


def download_and_display_m51_image():

    # Set the coordinates and image size for M51
    target_name = "M51"
    ra = 202.4697  # Right Ascension in degrees
    dec = 47.1953  # Declination in degrees
    image_size = 400  # Size of the image in pixels


    # Query the online database and download the optical image
    images = SkyView.get_images(position=f"{ra} {dec}",
                survey='DSS',
                coordinates='J2000', pixels=image_size)
    image = images[0]

    # Load the downloaded FITS file
    hdulist = image[0]
    image_data = hdulist.data


    # Display the image
    plt.figure(figsize=(8, 8))
    plt.imshow(image_data, cmap='gray')
    plt.title(target_name)
    plt.axis('off')
    plt.show()
```


M51

Colorimetry for Visualization

Want your plots

to look top notch?

https://seaborn.pydata.org/

https://seaborn.pydata.org/tutorial/color_palettes.html#general-principles-for-using-color-in-plots

See references at the bottom of this page:
https://medium.com/hipster-color-science/a-beginners-guide-to-colorimetry-401f1830b65a
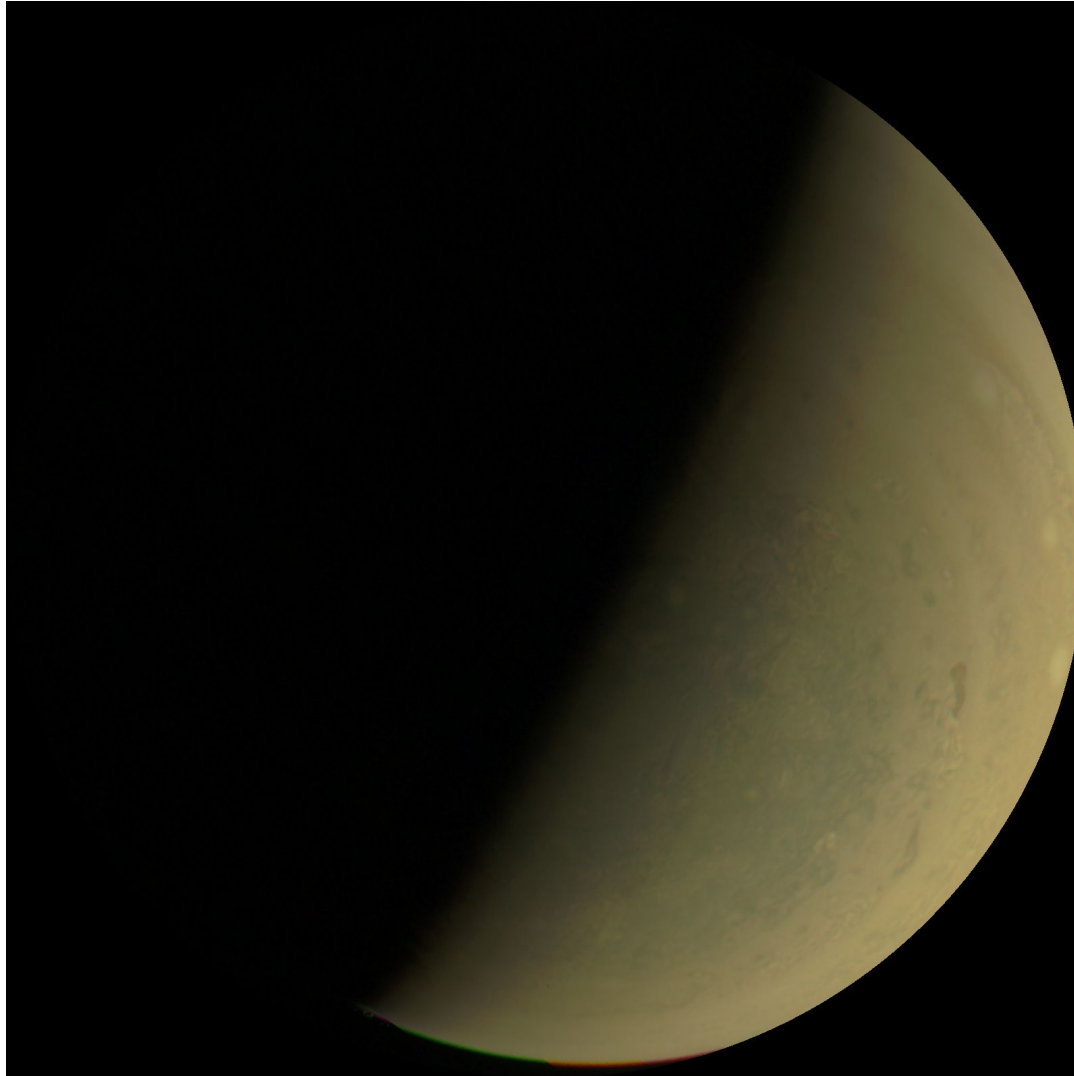
Talk by Dr. Michael Waters:
https://www.dropbox.com/s/7s9seplrnw3ea7p/Practical_Colorimetry_for_Scientific_Visualization_-_Michael_J_Waters_-_2021_3_14.pdf

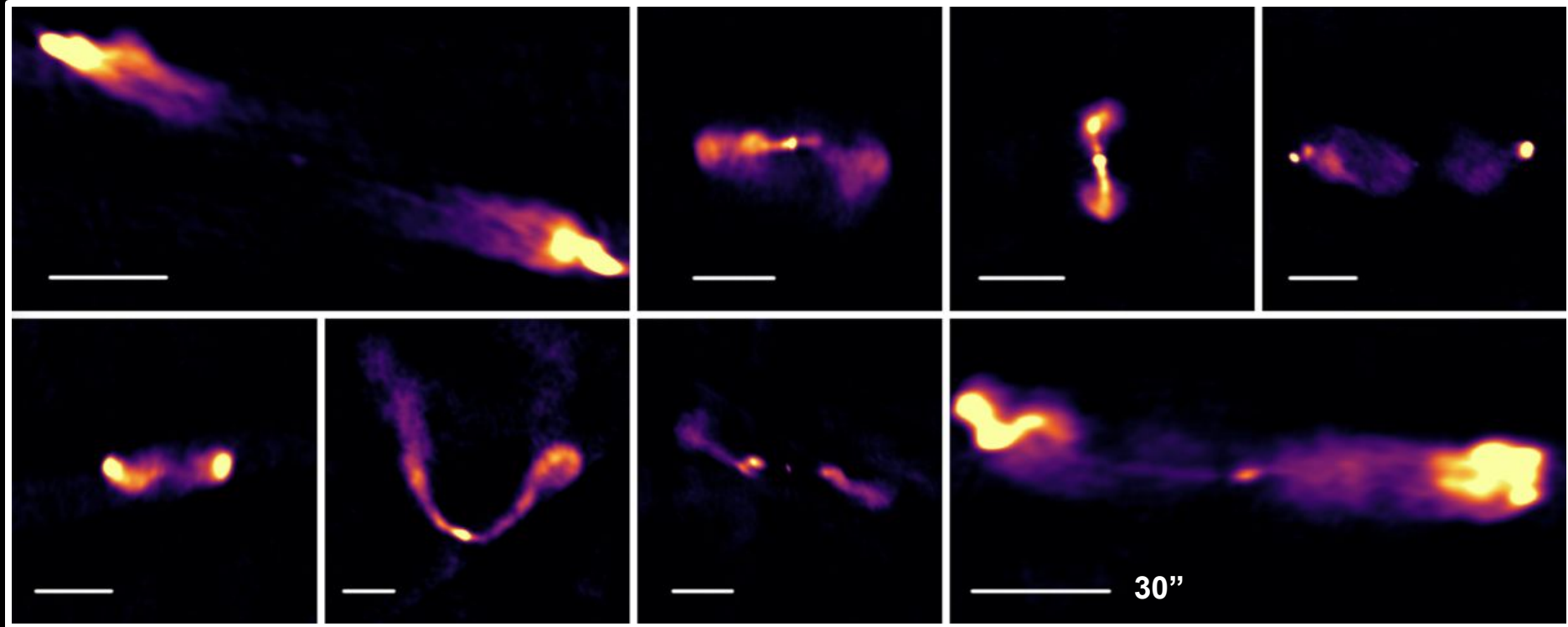**Recommendation:** Make your plots and graphics stand out for publication and presentation!
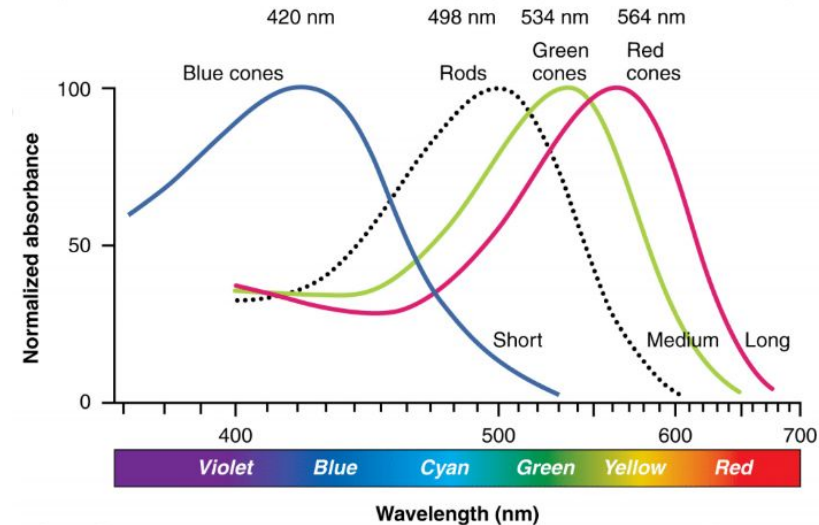
Another Example: Combine images from Juno
https://www.missionjuno.swri.edu/media-gallery/junocam

TensorFlow Example: VLASS Galaxy Classification

30"

**Machine Learning/Deep Learning
Neural Networks/Artificial Intelligence**

NVidia Deep Learning:

https://www.nvidia.com/en-us/deep-learning-ai/education/

PyTorch (Meta/Facebook):
https://pytorch.org/

Tensorflow (Google):

https://www.tensorflow.org/

Check out the NVidia GTC: https://www.nvidia.com/en-us/gtc/

Recommendation: *Please subscribe and sign up for these free tutorials and resources!*