All Things matplotlib

Getting the most out of your favorite
(but hated) Python plotting package.

by Taylor Hutchison

@aibhleog

General Overview

01 **the basics of matplotlib**
How to use it, what the defaults look like, and more!

02 **changing the defaults**
Style sheets, matplotlibrc files, or just adding lines of code.

03 **good plotting practices in astronomy**

04 **effective communication vs. poor construction**
We'll walk through some examples.

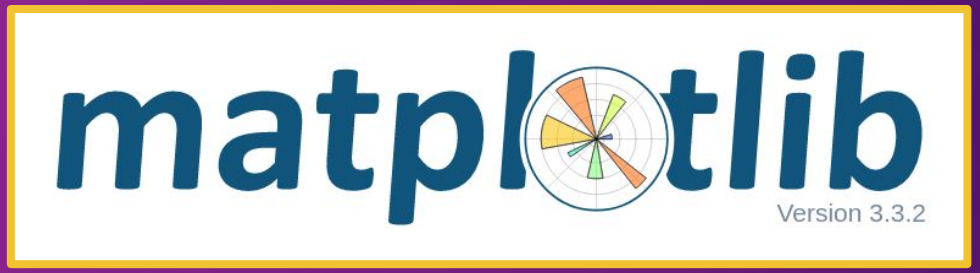
05 **summary & discussion**

06 **taking it a step further**
Modifying your .bashrc to play with plots!

01

basics of matplotlib

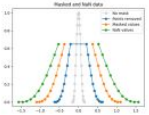
import matplotlib.pyplot as plt



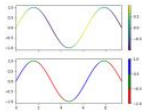
“Matplotlib is the brainchild of John Hunter (1968-2012), who, along with its many contributors, have put an immeasurable amount of time and effort into producing a piece of software utilized by thousands of scientists worldwide.”

- Visualization package with Python
- Hosted on GitHub (github.com/matplotlib/matplotlib)
- Open source, active developer community
- Several add-on toolkits also available
  - (mplot3d, axis helpers, etc.)
- Countless examples showcasing its functionality
- *What most astronomer use to make plots for their papers and presentations* (ignoring R & IDL)

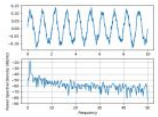
import matplotlib.pyplot as plt



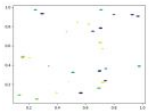
Plotting masked and NaN values



Multicolored lines



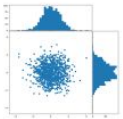
Psd Demo



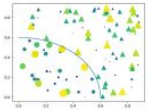
Scatter Custom Symbol

there are even more examples...

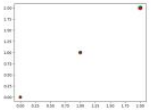
Scatter Demo2



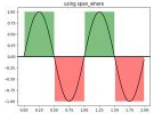
Scatter plot with histograms



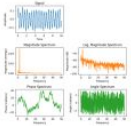
Scatter Masked



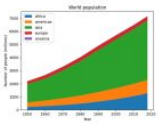
Scatter plot with pie chart markers



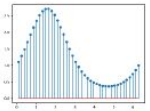
Using span_where



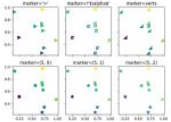
Spectrum Representations



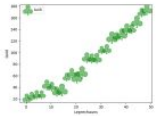
Stackplots and streamgraphs



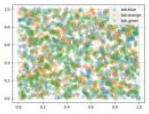
Stem Plot



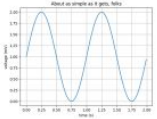
Marker examples



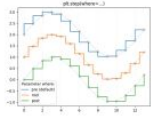
Scatter Symbol



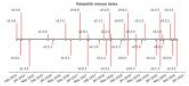
Scatter plots with a legend



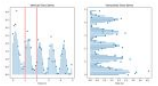
Simple Plot



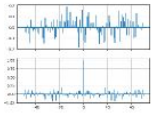
Step Demo



Creating a timeline with lines, dates, and text



hlines and vlines



Cross- and Auto-Correlation Demo

import matplotlib.pyplot as plt

Scatter plot with histograms

Show the marginal distributions of a s

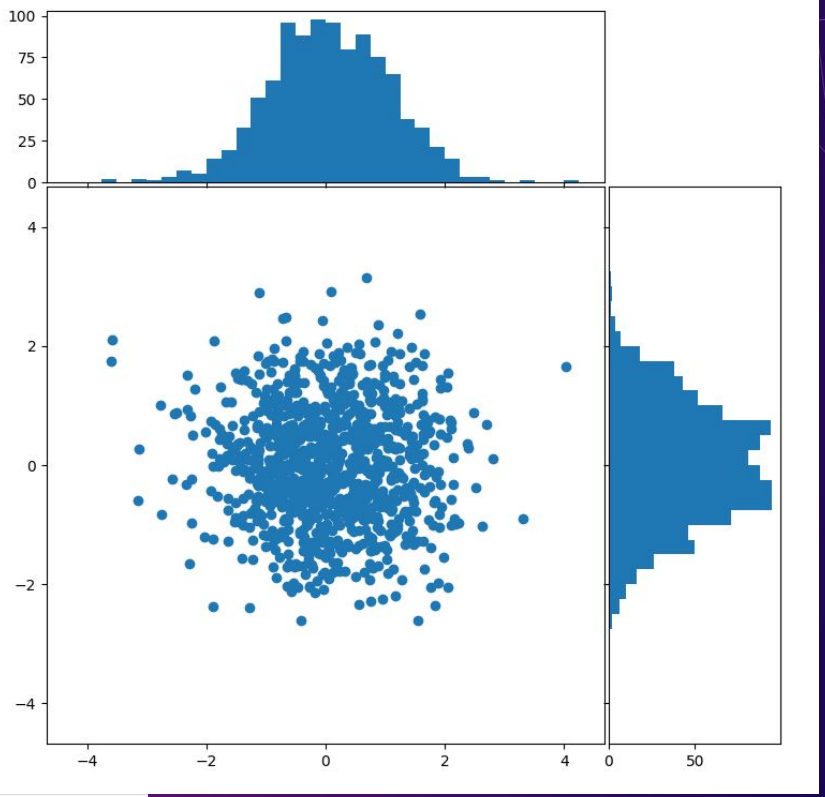
F

```
import numpy as np
import matplotlib.pyplot

# Fixing random state fo
np.random.seed(19680801)

# some random data
x = np.random.randn(1000
y = np.random.randn(1000


def scatter_hist(x, y, a
    # no labels
    ax_histx.tick_params
    ax_histy.tick_params

    # the scatter plot:
    ax.scatter(x, y)

    # now determine nice
    binwidth = 0.25
    xymax = max(np.max(n
    lim = (int(xymax/bin

    bins = np.arange(-li
    ax_histx.hist(x, bin
    ax_histy.hist(y, bin
```

Axes in figure coor

To define the axes positions, l
marginal axes share one dime

```
# definitions for the
left, width = 0.1, 0.6
bottom, height = 0.1,
spacing = 0.005


rect_scatter = [left,
rect_histx = [left, bo
rect_histy = [left + w

# start with a square
fig = plt.figure(figsi

ax = fig.add_axes(rect
ax_histx = fig.add_axe
ax_histy = fig.add_axe

# use the previously d
scatter_hist(x, y, ax,

plt.show()
```

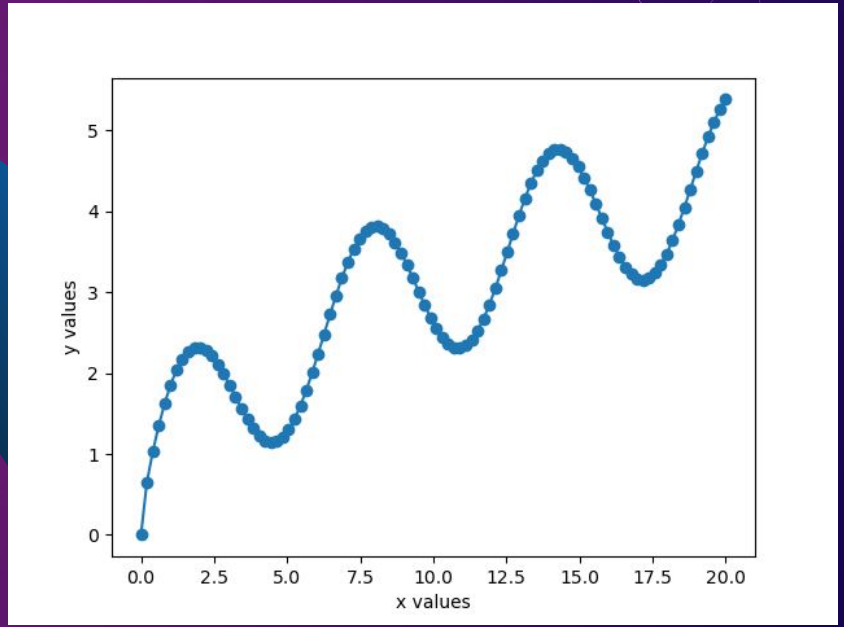


02

changing the defaults

the matplotlib defaults

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(0,20,100)
5 y = np.sin(x) + np.sqrt(x)
6
7 plt.figure()
8 plt.scatter(x,y,zorder=4)
9 plt.plot(x,y)
10 plt.ylabel('y values')
11 plt.xlabel('x values')
12 plt.savefig('test.png')
13 plt.close()
14
```

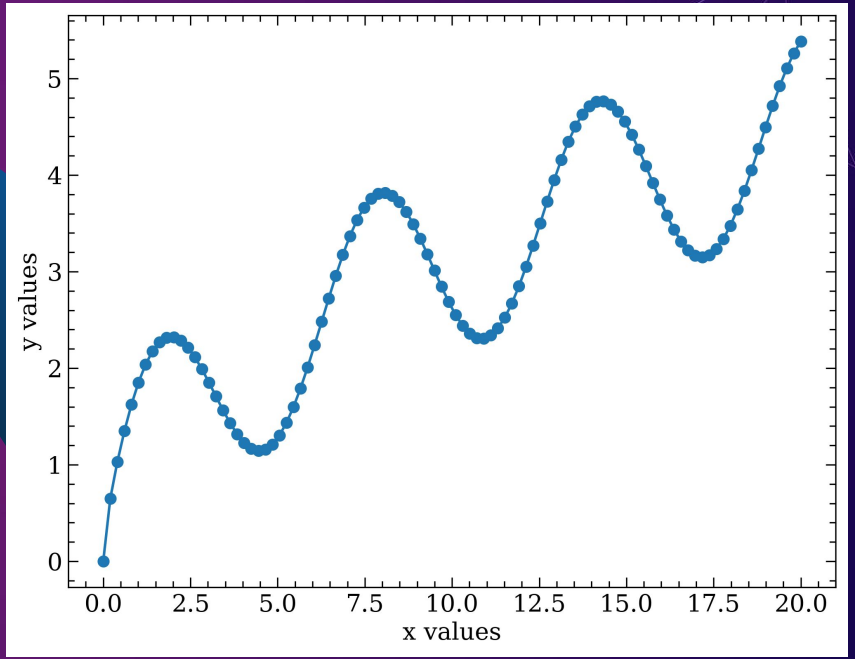


- tick marks outwards
- minor tick marks
- small font sizes
- large white borders

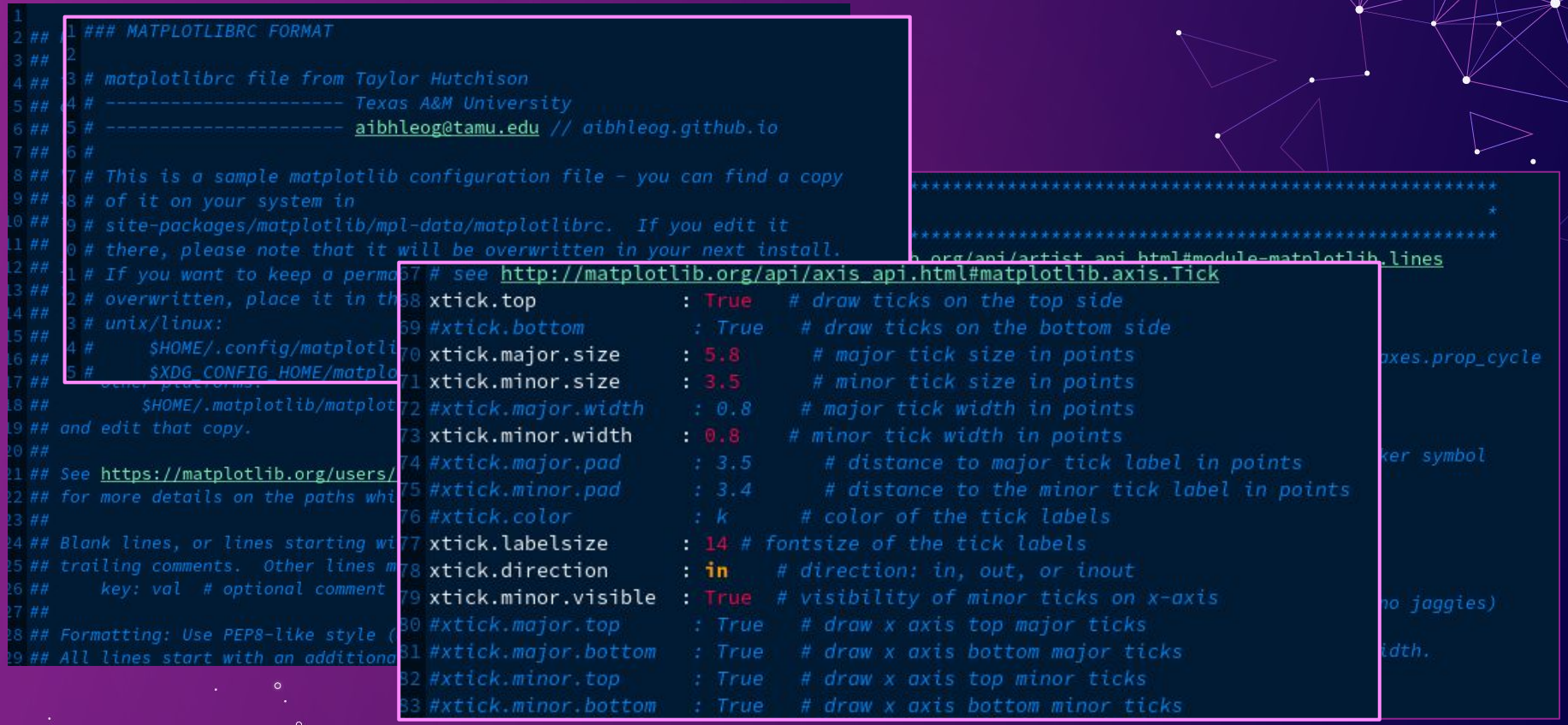
NOT the matplotlib defaults

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

x = np.linspace(0,20,100)
y = np.sin(x) + np.sqrt(x)

plt.figure()
plt.scatter(x,y,zorder=4)
plt.plot(x,y)
plt.ylabel('y values')
plt.xlabel('x values')
plt.savefig('test.png')
plt.close()
```

- tick marks inwards
- larger font sizes
- small white borders



1) the matplotlibrc file



2) style sheets

Style sheets are smaller lists of particular parameters you'd like to change (versus all of the available ones via the matplotlibrc file).

This can be useful for plots used in particular things, such as presentation plots, telescope proposal plots, etc., where you may want or need a different style than the general defaults you like.

```
axes.titlesize : 24
axes.labelsize : 20
lines.linewidth : 3
lines.markersize : 10
xtick.labelsize : 16
ytick.labelsize : 16
```

can save to presentation.mplstyle

plt.style.use('presentation.mplstyle')

```
In [2]: import matplotlib.pyplot as plt

In [3]: print(plt.style.available)
['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast', '
fivethirtyeight', 'ggplot', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind
', 'seaborn-dark', 'seaborn-dark-palette', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-m
uted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-
talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

3) hard-coding it (not advised)

You could also hard-code the things you want:

```
import matplotlib.pyplot as plt

f, ax = plt.subplots(1,1)

[plotting code]

ax.tick_params(which='both', axis='both', direction='in')
ax.yaxis.set_ticks_position('both')
ax.xaxis.set_ticks_position('both')
ax.minorticks_on()

ax.tight_layout()
ax.show()
ax.close()
```

For example, but
I don't advise this....

4) a little less hard-coding it (still not advised)

This way of hardcoding would apply to an entire script or Jupyter notebook, versus just one plot... but still... I wouldn't advise it.

```
import matplotlib as mpl
```

```
mpl.rcParams['lines.linewidth'] = 2
mpl.rcParams['lines.linestyle'] = '--'
plt.plot(data)
```

Mostly just because if you plan to do this for every plot you make, you may as well set the defaults and not have to worry about this.

03

good plotting practices in astronomy

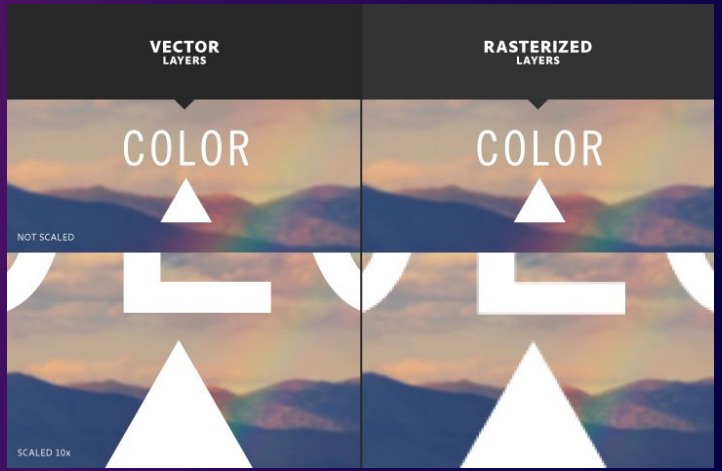
in general,

Tick marks facing inwards
Minor ticks on
Not made in excel

Clear & understandable

- Not too much going on
- Large font size for all text, tick marks, & legends
- Avoid awful colors
- Thicken lines
- Add edge colors to scatter points
  - or add an “alpha” to them (semi-transparent)
- (*Taylor’s pet peeve*) make the plot font match the font of the paper or presentation (if you can)

Needs to be vectorized, however you can rasterize certain things in a plot so the PDF loads faster (like dense scatter plots)



04

effective communication vs. poor construction

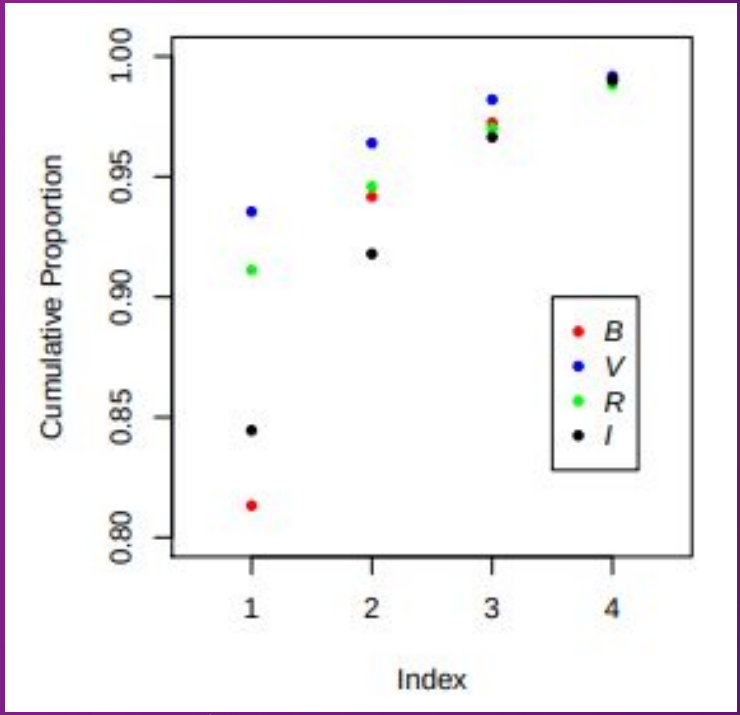
We're going to look at some examples, both good & bad

but this is going to be an instructive tour, not one that shames any particular person.

plotting effectively is a skill, with many left untrained by no fault of their own.

there's ALWAYS room to improve!

examples



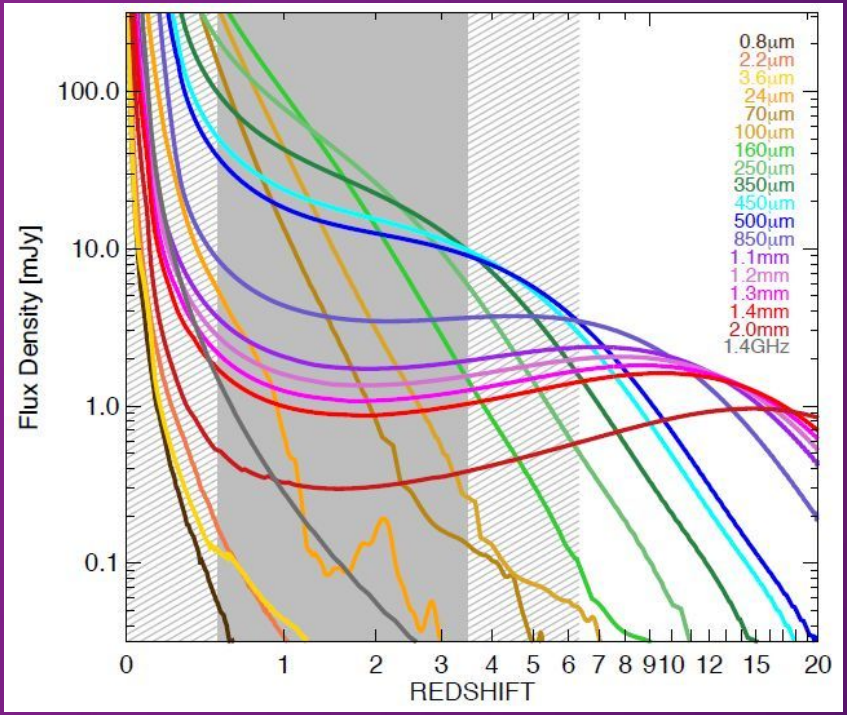
ways to improve

- font sizes
- outward tick marks
- no minor tick marks

- THE COLOR OF THE POINTS
  - B ▲
  - V ●
  - R ◆
  - I ■

(shapes make it easier to follow & colorblind friendly, the coloring makes more sense with the λ's)

examples



ways to improve

- make the legend 2 columns (but not absolutely necessary)
- change line types to be more colorblind-friendly

examples

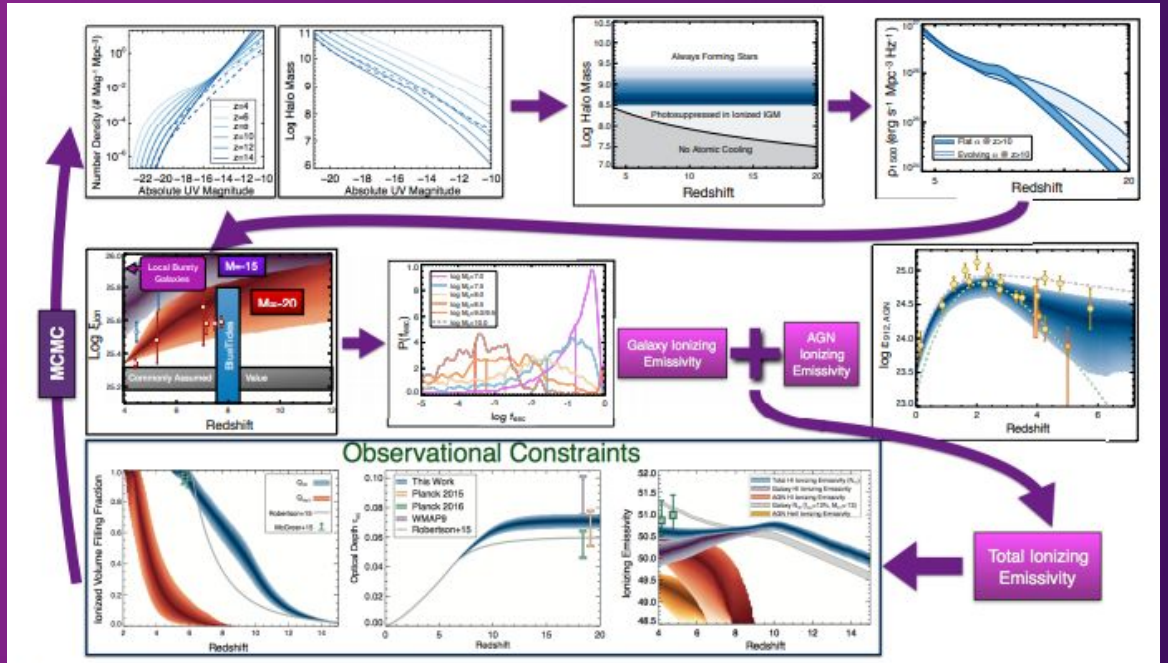
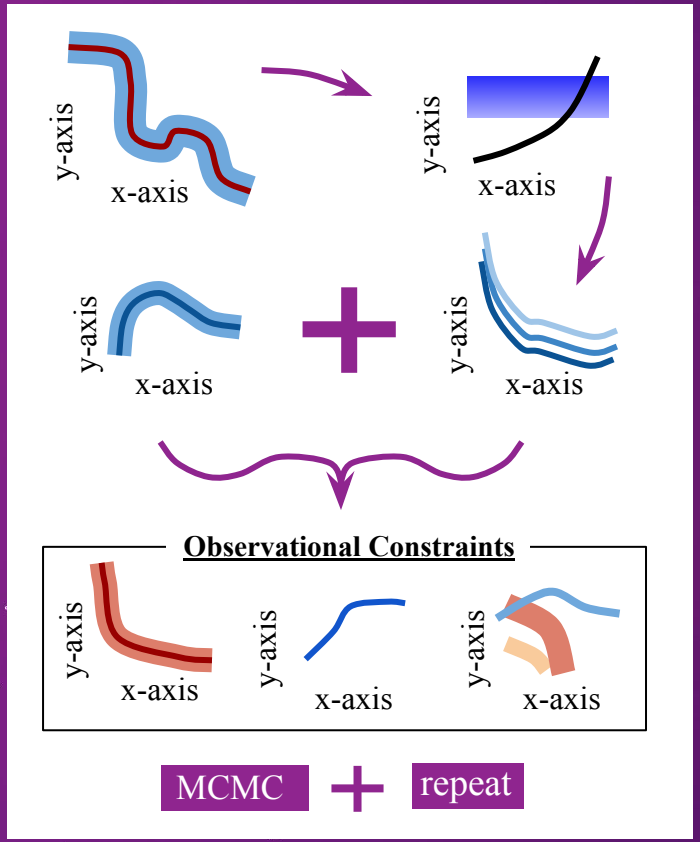


Figure 4. A visual description of our Markov Chain Monte Carlo procedure for constraining the posteriors on our free parameters, described in full in § 3.1. All figures appear full-size elsewhere in the paper.

ways to improve

- space things out
  - this image takes up ½ a page in the paper, could spread it out to fit 1 page
- turn plots into more of a graphical design (no axes)
- remove legends
- make labels larger
- need to vectorize figure

examples



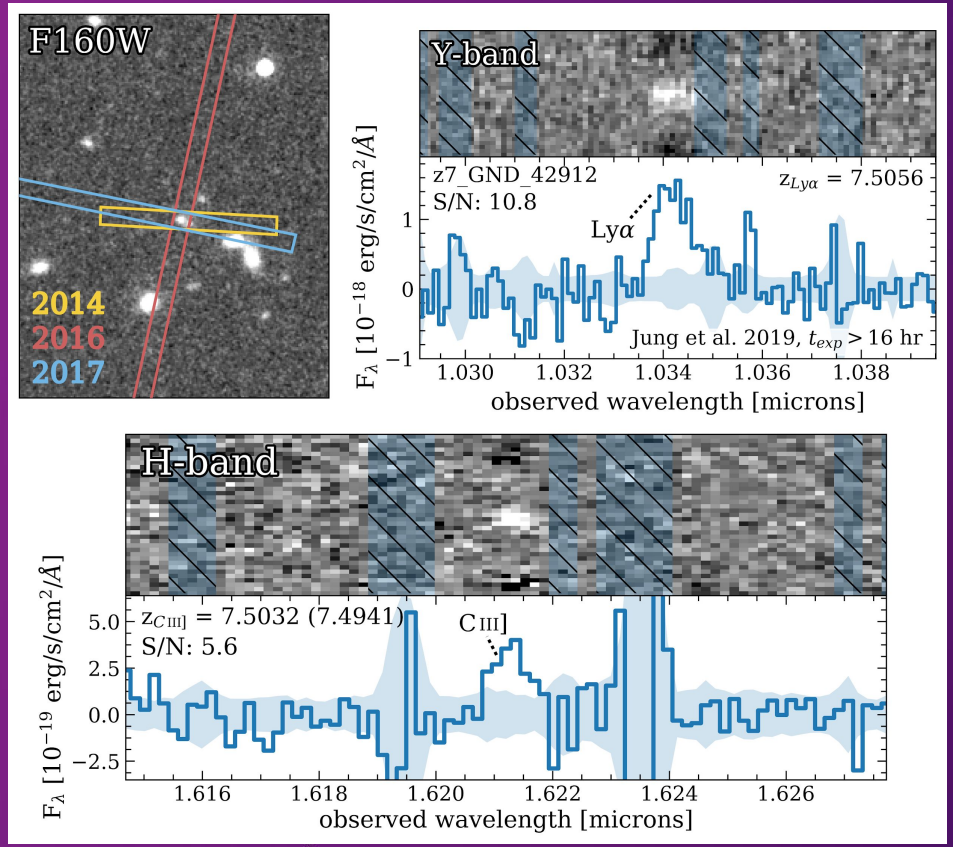
Possible solution, or could do something else....

ways to improve

- space things out
  - this image takes up ½ a page in the paper, could spread it out to fit 1 page
- turn plots into more design graphical (remove axes)
- remove legends
- make labels larger

examples

one of mine



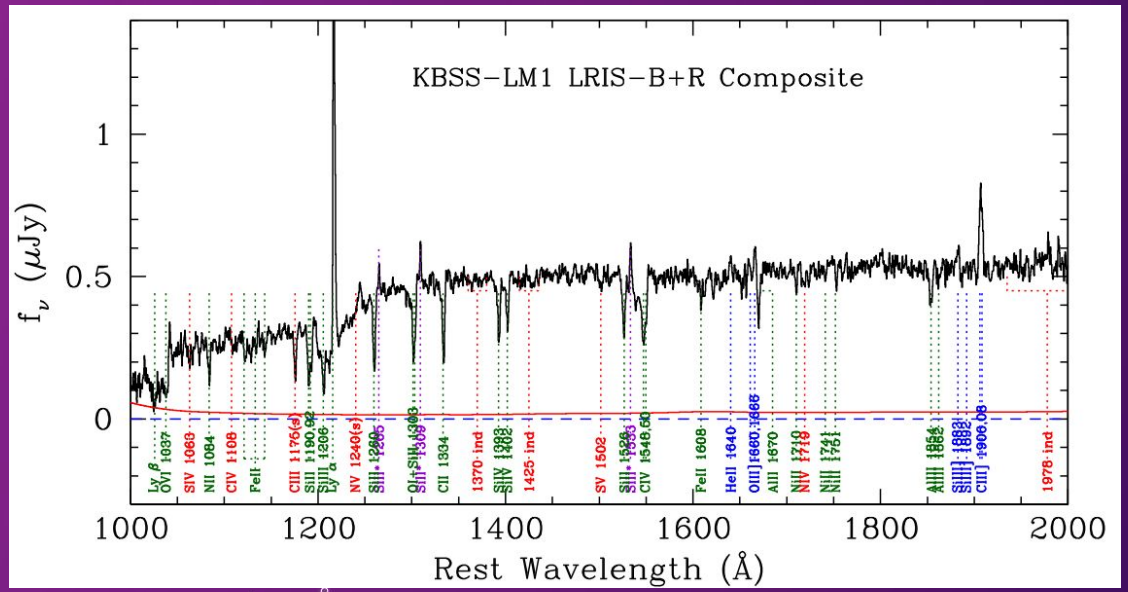
ways to improve

- color the “F106W”, “Y-band”, and “H-band” to stand out
- less text?
- other thoughts?

- suggested during talk:
  - add something to highlight the emission lines more
  - make the red color pop more in the top left subplot

examples

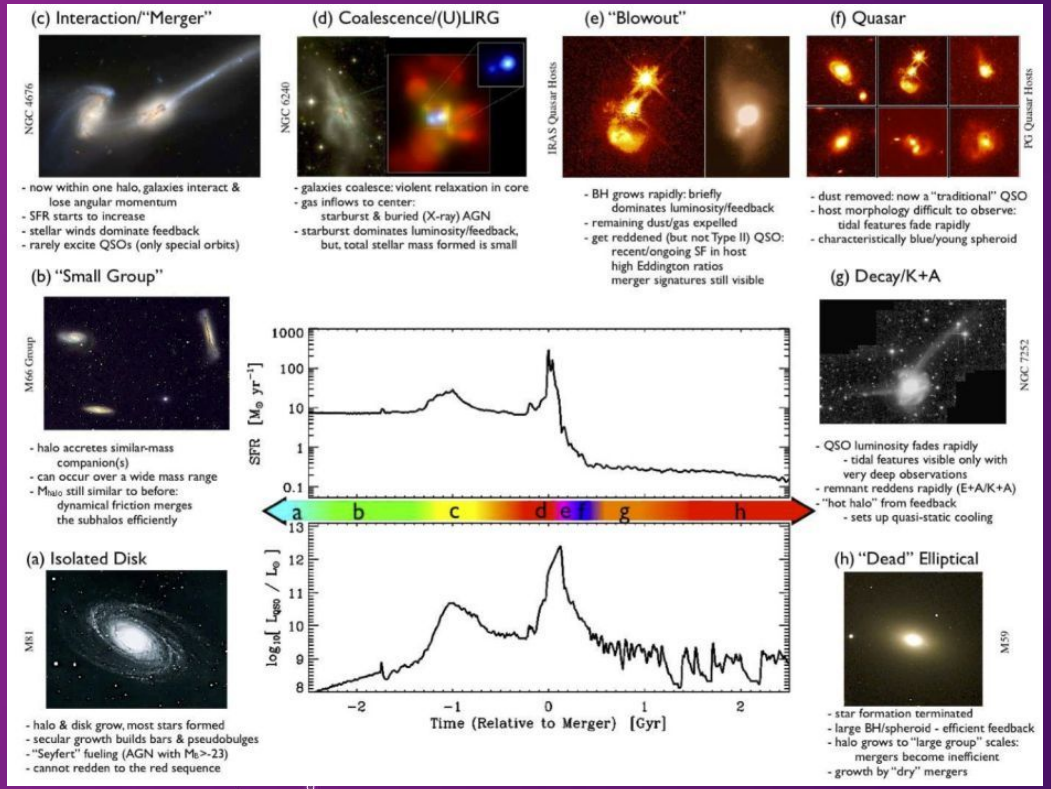
As a note, this plot is the width of the page in the paper, which is good. Any smaller, and more things would need to change.



ways to improve

- add legend for the line colors
- changing line types for the different line colors (colorblind)

examples



ways to improve

- make text slightly larger
- label for the colorbar
- Spread things out
  - could move two to the bottom and have even spacing around the central figure

As you can see, there are a lot of different ways to convey information...

but some are *far more* effective than others.

05

summary & discussion

SUMMARY

Think through your plots

- How do you want to convey the information?
- How can you do this clearly?
- What kind of coding would this require?
- Get second opinions on your plots (if you're not sure)

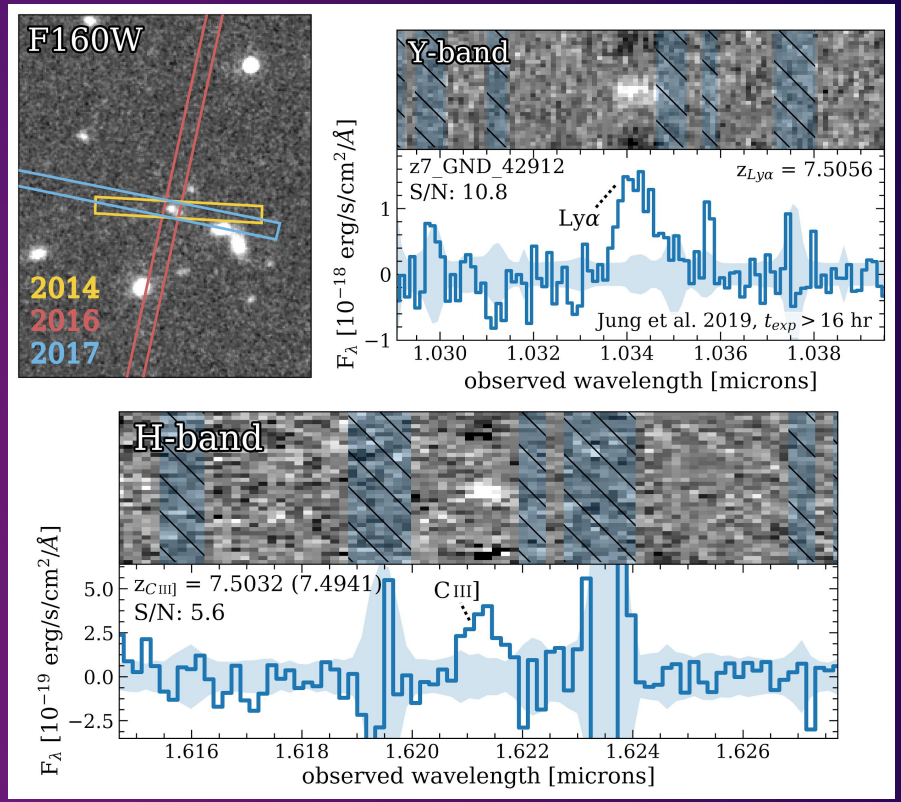
Coding your plots

- Play around with stylesheets or a matplotlibrc file
- Don't make yourself suffer through the basic defaults
  - Save yourself from typing the same code for every plot, when you could just change the defaults
- stackexchange / stackoverflow is your friend
- Take advantage of the plotting knowledge of your peers!
  - There's always more to learn
- Try to challenge yourself with your plotting (get creative!)

DISCUSSION

There are many ways to make your plots fancier

- multiple subplots

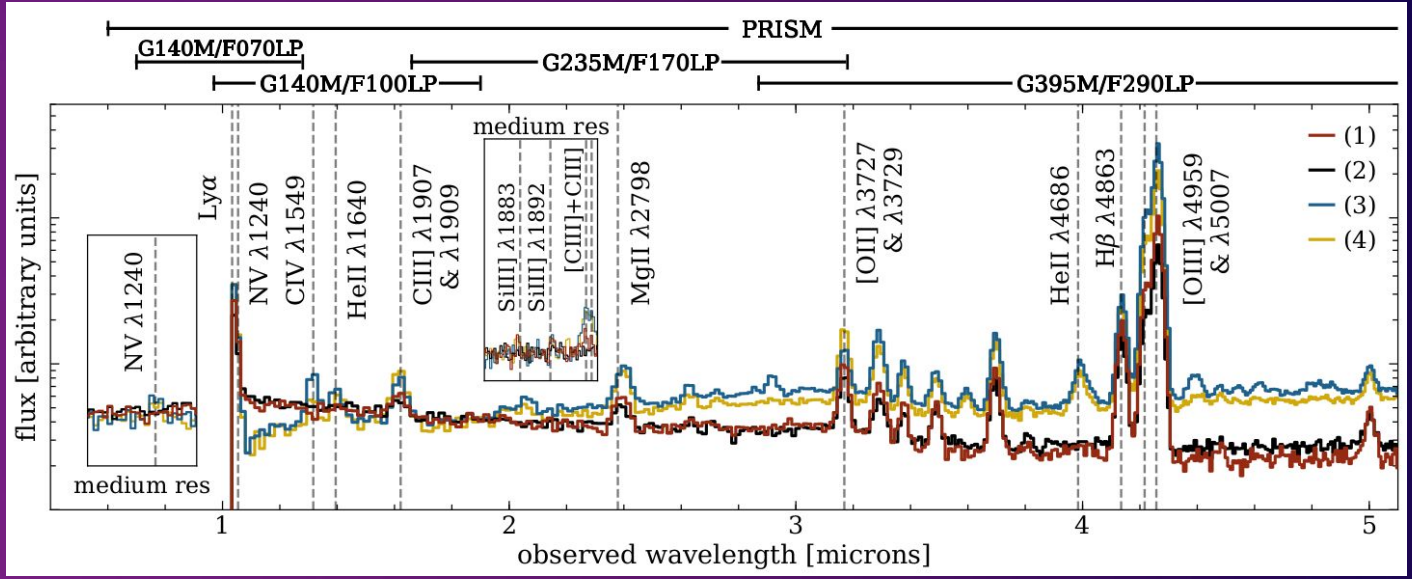


some examples from my coding

DISCUSSION

There are many ways to make your plots fancier

- multiple subplots

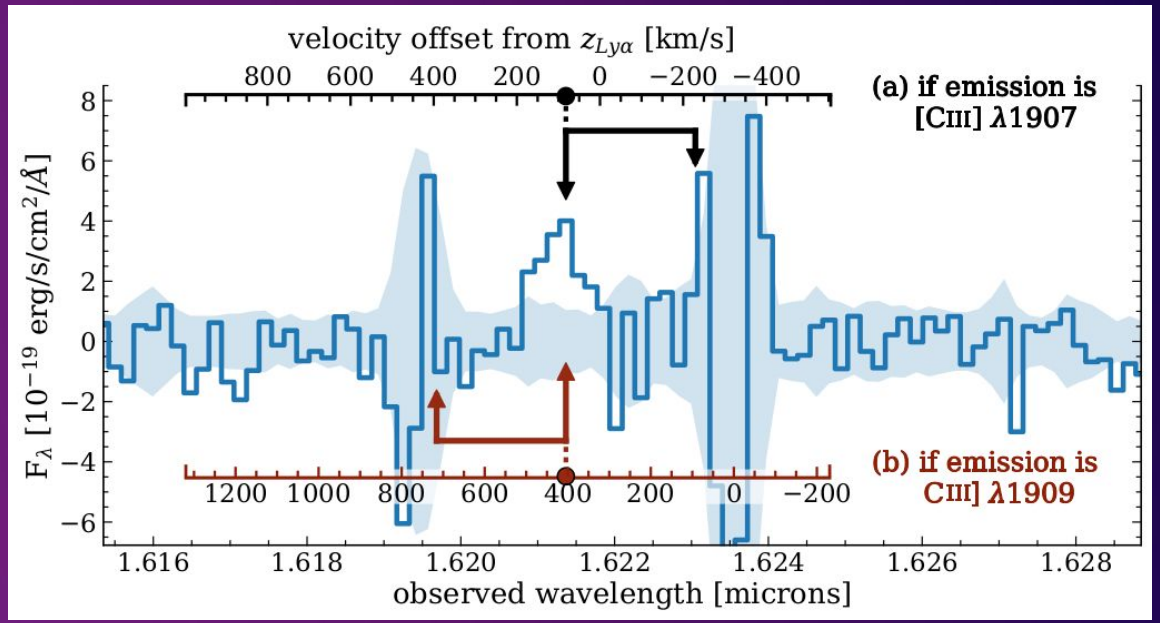


some examples from my coding

DISCUSSION

There are many ways to make your plots fancier

- multiple subplots
- inset subplots

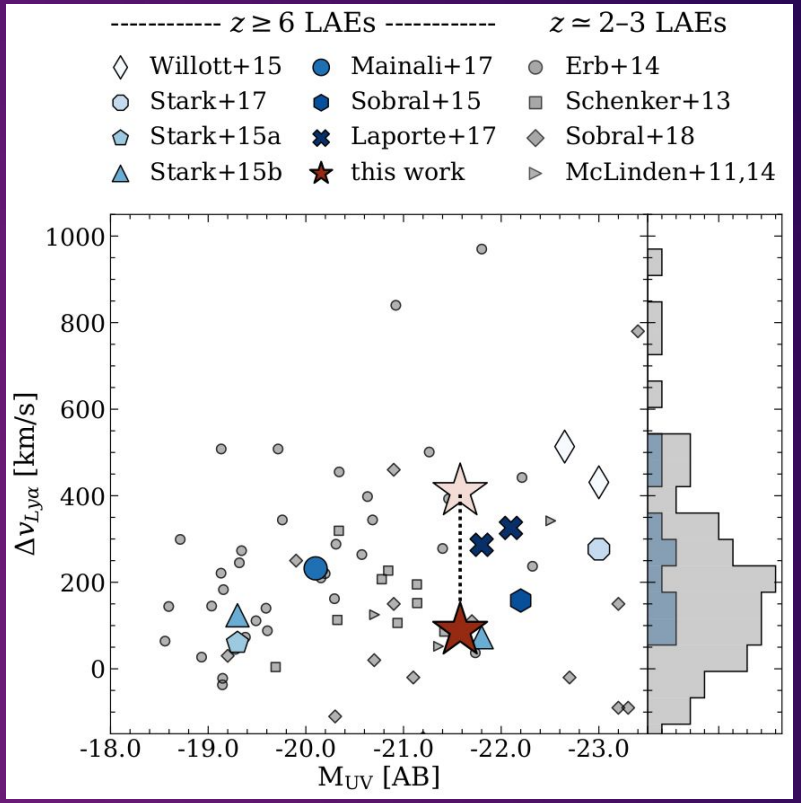


some examples from my coding

DISCUSSION

There are many ways to make your plots fancier

- multiple subplots
- inset subplots
- moving legends outside the plot



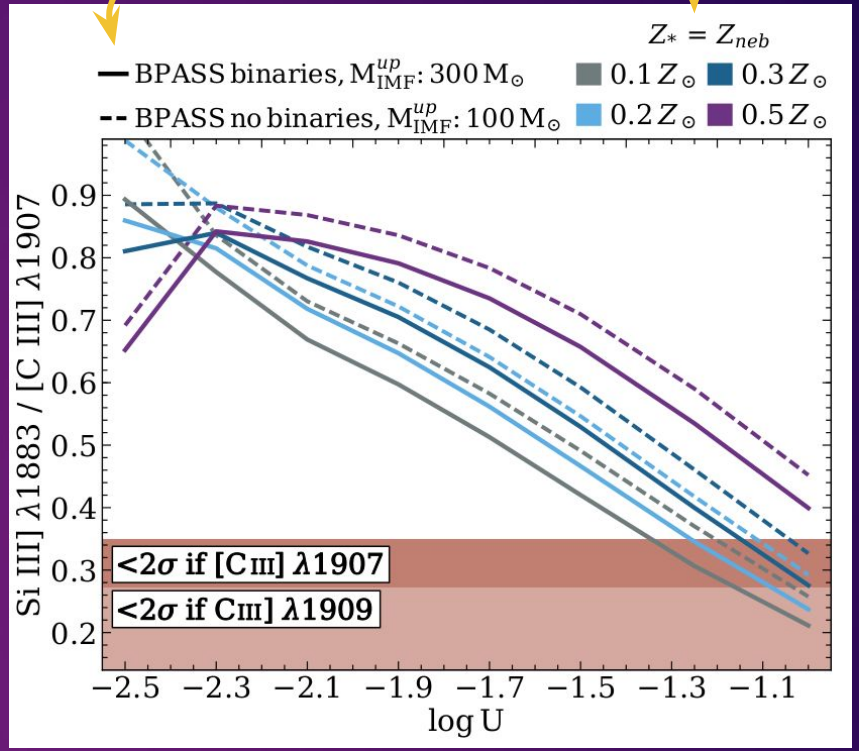
some examples from my coding

DISCUSSION

two diff. legends!

There are many ways to make your plots fancier

- multiple subplots
- inset subplots
- moving legends outside the plot



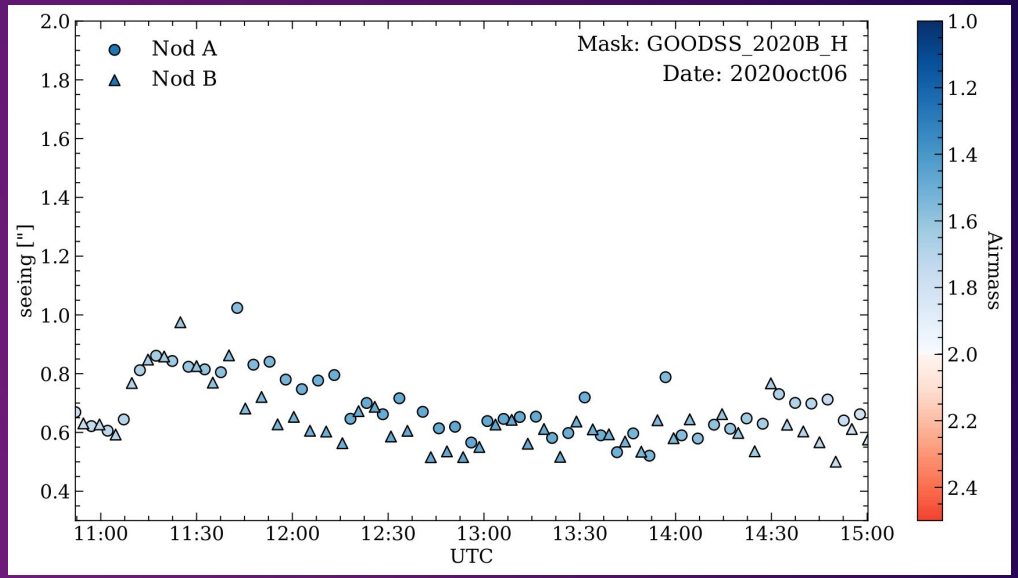
some examples from my coding

DISCUSSION

There are many ways to make your plots fancier

- multiple subplots
- inset subplots
- moving legends outside the plot
- anchoring colorbars to specific numbers

anchored to white being at airmass=2

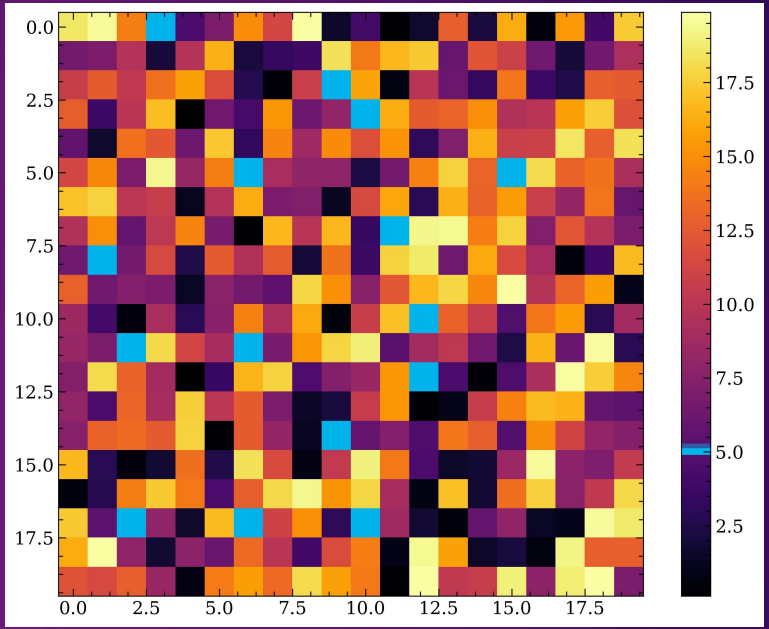


some examples from my coding

DISCUSSION

There are many ways to make your plots fancier

- multiple subplots
- inset subplots
- moving legends outside the plot
- anchoring colorbars to specific numbers



some examples from my coding

DISCUSSION

There are many ways to make your plots fancier

- multiple subplots
- inset subplots
- moving legends outside the plot
- anchoring colorbars to specific numbers
- & more!

More resources:

color-blindness.com/coblis-color-blindness-simulator/

matplotlib.org

For my plots shared today, most of the code is available on my GitHub: github.com/aibhleog/plotting-playground
github.com/aibhleog/Quick-Tools-For-The-Observational-Astronomer

06

taking it
a step further

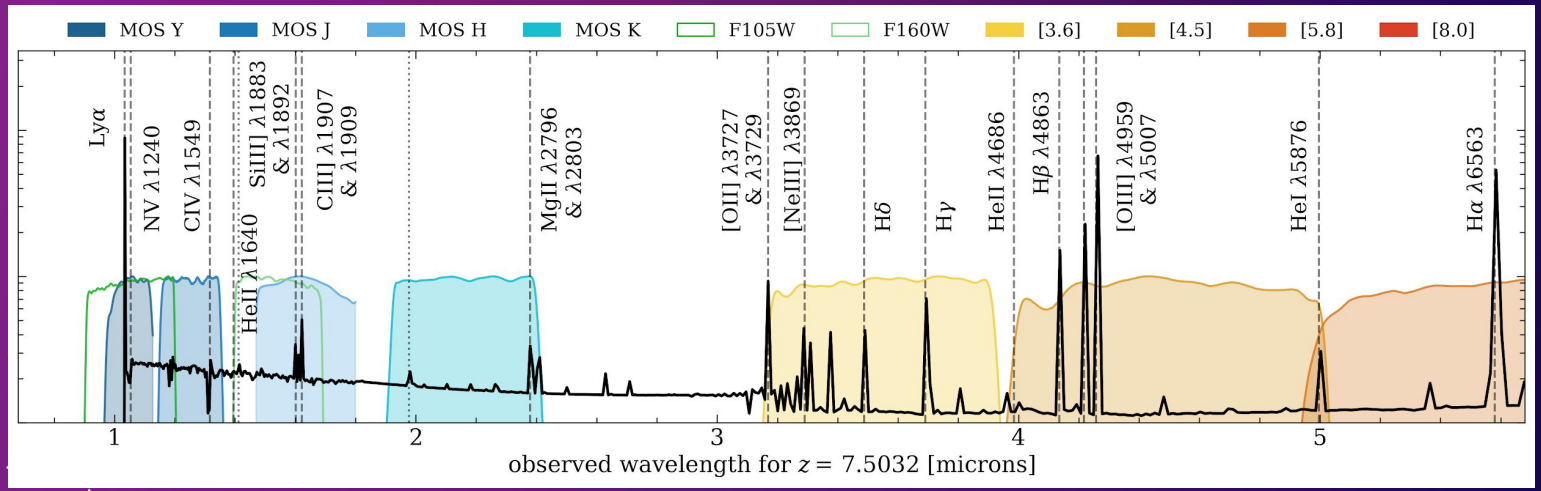
involving your .bashrc

You can add aliases to your .bashrc file to run scripts, if you have ones that you'd like to run with inputs that are useful for a specific purpose.

For example, here's one I have in my .bashrc:

```
# alias for seeing what lines can pop up in NIR and IR bands given a z
alias zlines='python /home/aibhleog/Desktop/keckData/information/scripts/bandpass_zlines.py'
```

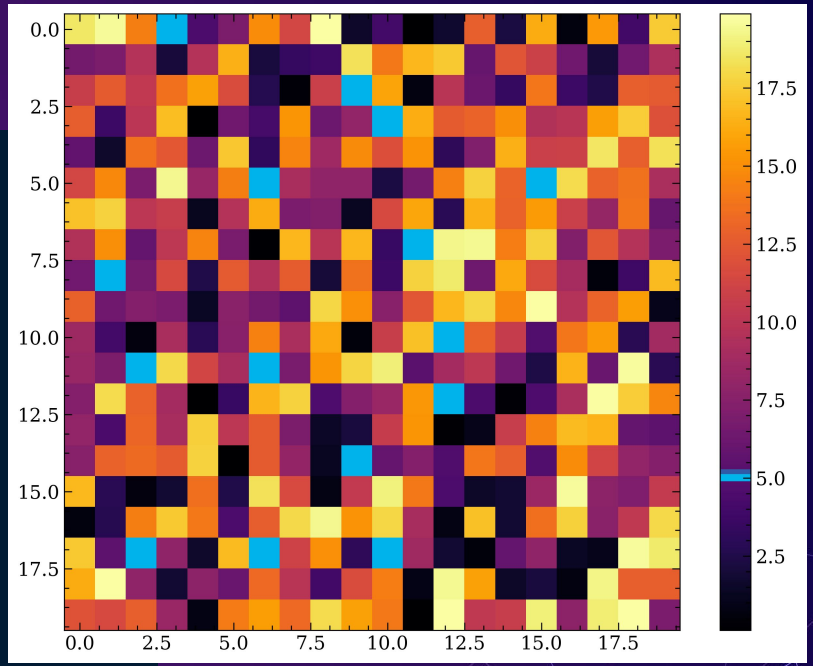
in terminal: `[aibhleog@earth ~]$ zlines 7.5032`



THANKS!

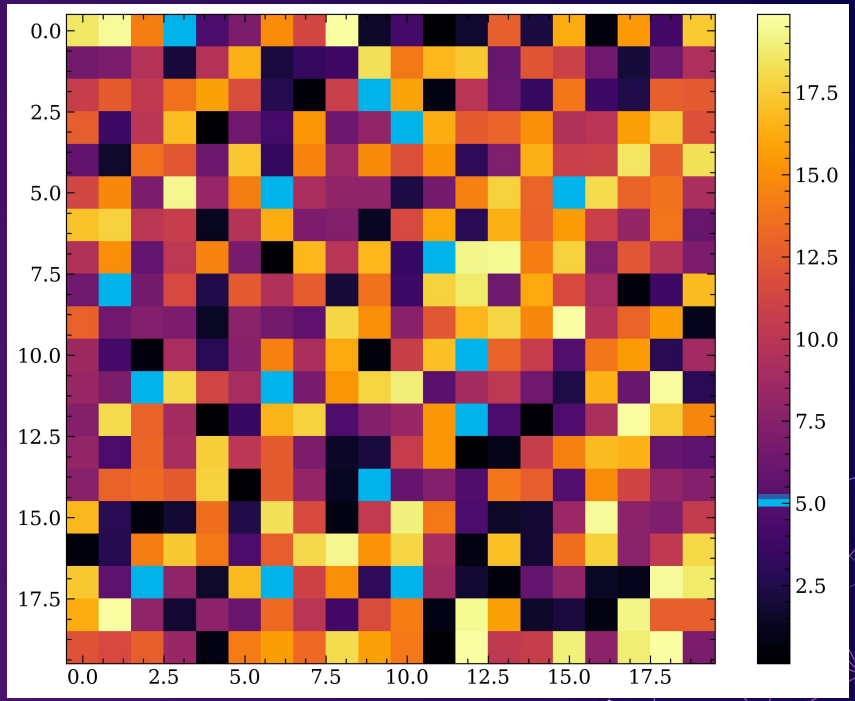
Examples with code #1

```python
def make_colormap(cen,dmin,mmax,size):
    # coming up with a range that helps us figure out where the color is
    t = np.linspace(dmin,dmax,size)
    s = np.linspace(0,1,size)
    n = size*10

    x = cen/(dmax-dmin)
    print('x: %s, Min: %s, Max: %s\n'%(x,dmin,dmax))

    lower = plt.cm.Reds(np.linspace(0.6,0,n*round(x-0.01*x,2)))
    #focus = plt.cm.rainbow(np.ones(3)*x)
    upper = plt.cm.Blues(np.linspace(0,1,n*round(1-(x+0.01*x),2)))
    #colors = np.vstack((lower, focus, upper))
    colors = np.vstack((lower, upper))
    tmap = matplotlib.colors.LinearSegmentedColormap.from_list('taylor', colors)
    return tmap

class FixPointNormalize(matplotlib.colors.Normalize):
    """
    Inspired by https://stackoverflow.com/questions/20144529/shifted-colorbar-matplotlib
    Subclassing Normalize to obtain a colormap with a fixpoint
    somewhere in the middle of the colormap.

    This may be useful for a `terrain` map, to set the "sea level"
    to a color in the blue/turquise range as shown in example:
    https://stackoverflow.com/questions/40895021/python-equivalent-for-matlabs-demcmap-elevation-appropriate-
colormap
    """
    def __init__(self, vmin=None, vmax=None, fixme=5, fixhere=0.26, clip=False):
        # fixme is the fix point of the colormap (in data units)
        self.fixme = fixme
        # fixhere is the color value in the range [0,1] that should represent fixme
        self.fixhere = fixhere
        matplotlib.colors.Normalize.__init__(self, vmin, vmax, clip)

    def __call__(self, value, clip=None):
        x, y = [self.vmin, self.fixme, self.vmax], [0, self.fixhere, 1]
        return np.ma.masked_array(np.interp(value, x, y))
```



Examples with code #1

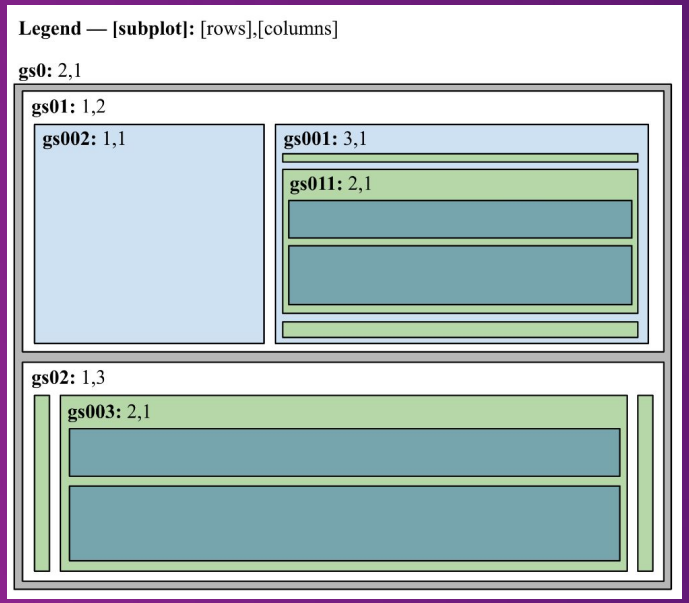
```
# making fake data
data = np.random.rand(20,20)
scale = 85
data *= scale
data[(data>4.7)&(data<5.3)] = 35.
print('There should be %s squares that are blue\n'%len(data[np.where(data == 5.)]))

# plotting data using implot to play with color maps
cmap = plt.get_cmap('inferno')
colors = cmap(np.linspace(0,1, len(data)+2))

dmax = max(data[0])
dmin = min(data[0])
cen = 35
x = cen/(dmax-dmin)

tmap = make_colormap(cen,dmin,dmax,len(data))
norm = FixPointNormalize(fixme=cen,fixhere=x,vmin=dmin,vmax=dmax)

plt.figure(figsize=(9,7))
co = plt.imshow(data, cmap=tmap)#, norm=norm)
cb = plt.colorbar(co, ax=plt.gca())

plt.tight_layout()
plt.savefig('test.png')
plt.close('all')
```

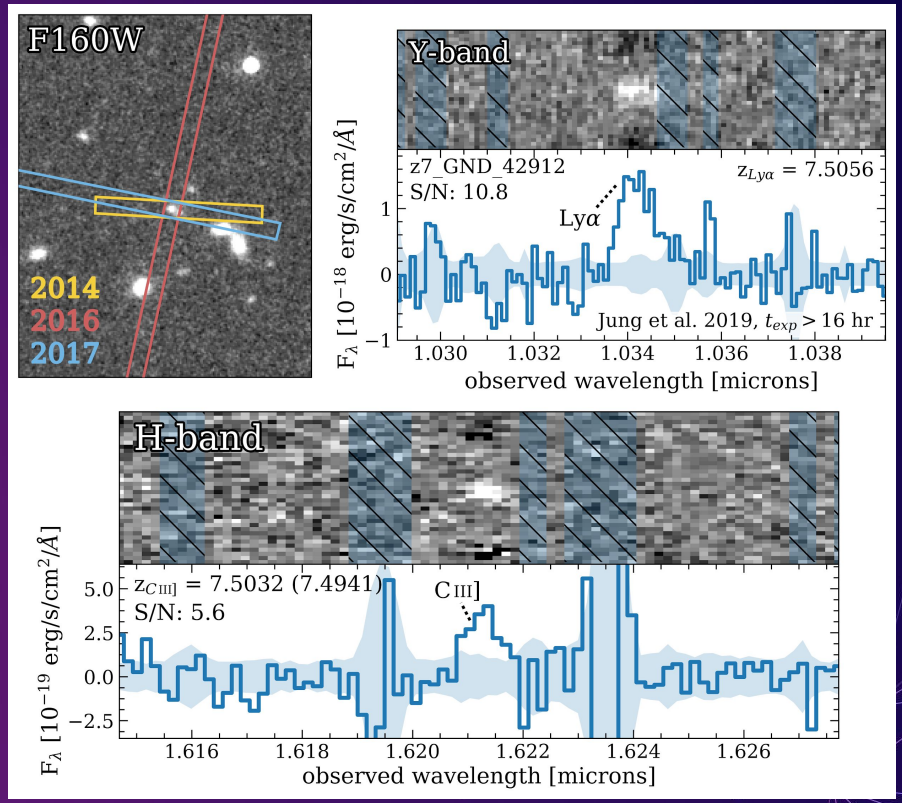


Examples with code #2

This was was actually several subplots & sub-subplots all used to tweak the 3 plots *exactly* where I wanted them.

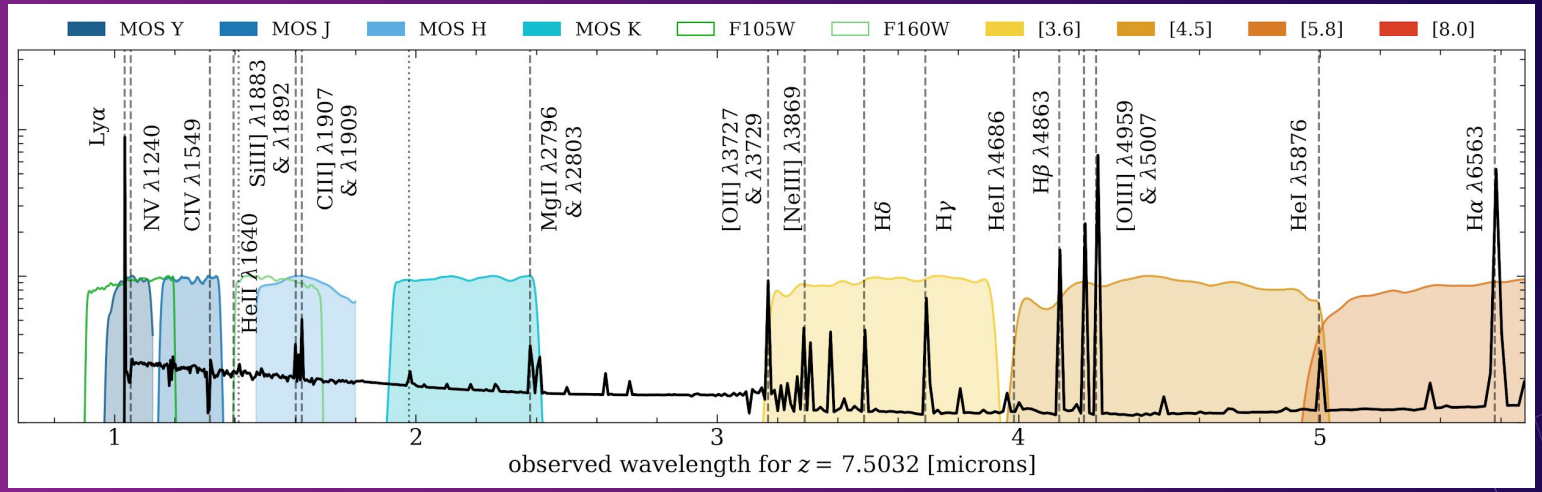
Here's the layout of the subplots:





Link to code:
github.com/aibhleog/plotting-playground

Examples with code #3

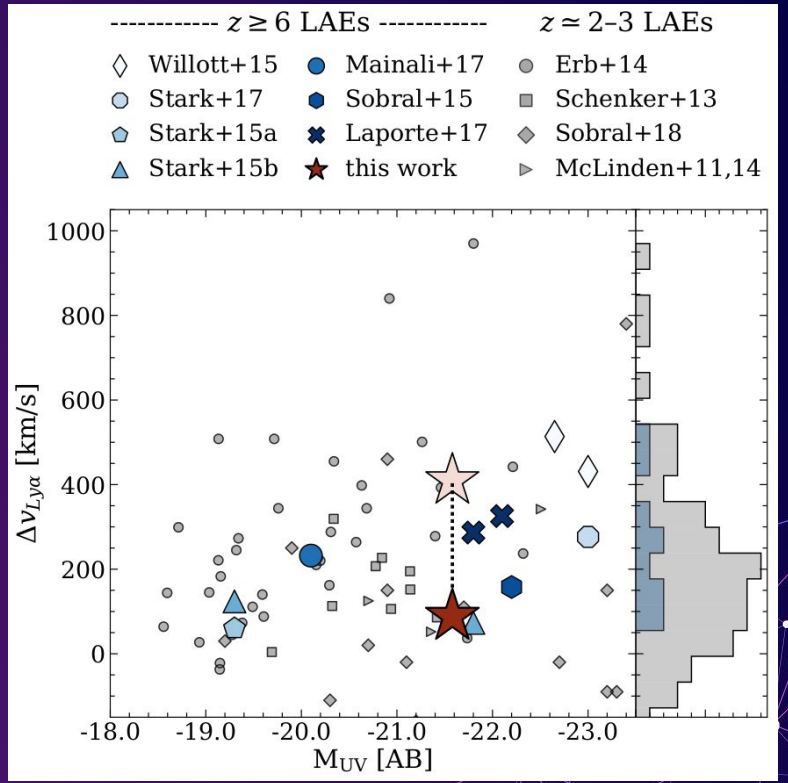


I've linked the code for this one below. In addition, the code is written as a script so that you can run it from the terminal as an alias (see part 6 of the slides for example).

Link to code:
github.com/aibhleog/plotting-playground

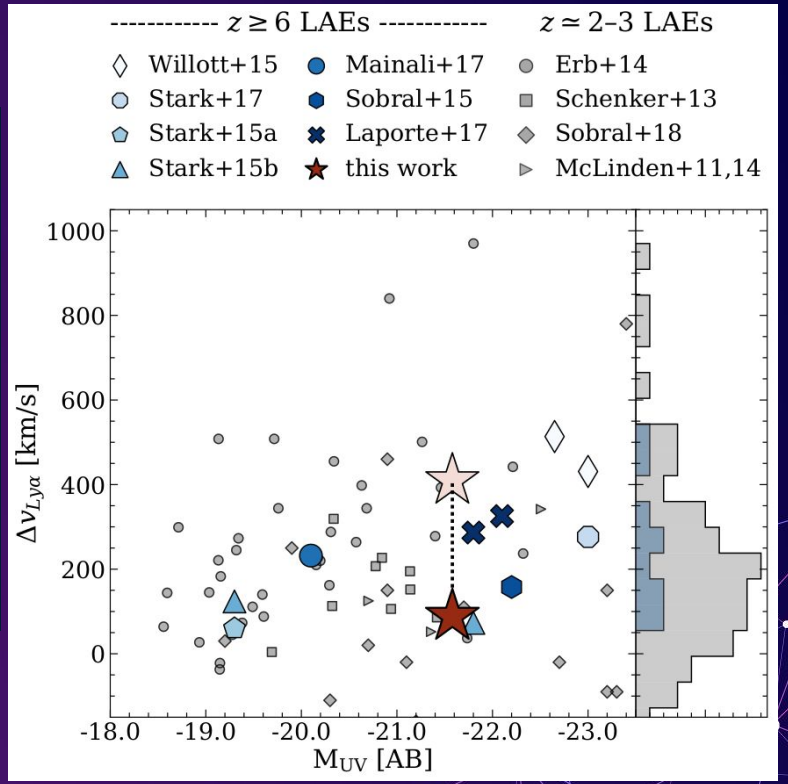
Examples with code #4

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

# code to read in tables and prep for plotting
[.................]


# plotting
plt.figure(figsize=(9,7))
gs1 = gridspec.GridSpec(1,2,width_ratios=[4,1]) # gridspec is your friend
gs1.update(wspace=0.0,hspace=0.0) # zero space between subplots

ax = plt.subplot(gs1[0]) # first subplot

ax.scatter(xvalues,yvalues,edgecolor='k',label='paper label 1')
ax.scatter(more_xvalues,more_yvalues,edgecolor='k',marker='s',label='paper label 2')
ax.scatter(even_more_xvalues,even_more_yvalues,marker='*',color='#962A13',lw=1.5,edgecolors='k',label='this work')
# [... more ax.scatter lines added ...]


# wanted my xaxis to show first decimal place for all tick labels
from matplotlib.ticker import FormatStrFormatter
ax.xaxis.set_major_formatter(FormatStrFormatter('%.1f'))

ax.legend(fontsize=18,ncol=3,handletextpad=0.22,columnspacing=0.6,bbox_to_anchor=(1.28,1.348))
# adding text as labels for the legend that's anchored outside the plot
ax.text(-18.0,1470,'------------ $z\geq6$ LAEs ------------',fontsize=20.5)
ax.text(-22.5,1470,'$z\simeq2\endash3$ LAEs',fontsize=20.5)

ax.set_xlim(-18,-23.5)
ax.set_ylim(-150,1050)
ax.set_xlabel(r'M$_{\rm UV}$ [AB]',fontsize=19.5)
ax.set_ylabel(r'$\Delta v_{Ly\alpha}$ [km/s]',labelpad=0,fontsize=20)
ax.tick_params(labelsize=18) # increasing size of tick labels
```

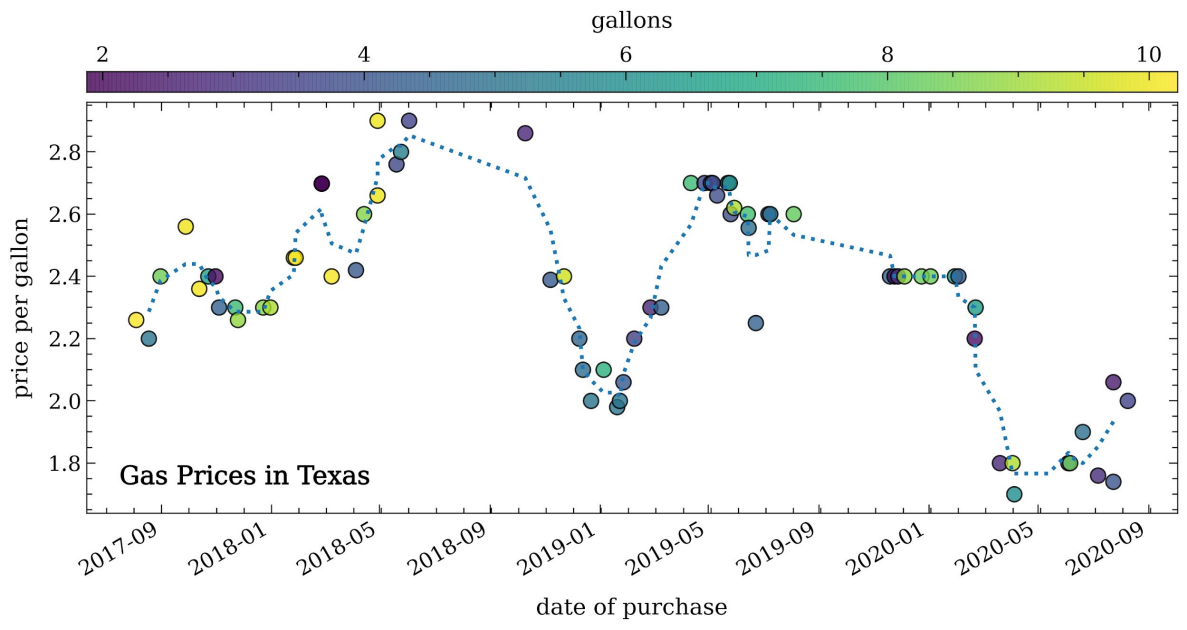


Examples with code #4

```
# ------------ histogram sideplot ------------------------------- #
ax2 = plt.subplot(gs1[1]) # second subplot

# predefining the bins (because I had multiple datasets)
bins=np.histogram(np.hstack((yvalues,more_yvalues,even_more_yvalues)), bins=20)[1]

# plotting hist twice: once with the fill color, once with just the outline
ax2.hist(lowz,bins,orientation='horizontal',alpha=0.4)
ax2.hist(lowz,bins,orientation='horizontal',color='k',histtype='step')

ax2.set_ylim(-150,1050)
ax2.set_yticklabels([])
ax2.set_xticklabels([])
# ------------------------------------------------------------- #

plt.savefig('/path/to/file.pdf',dpi=200)
plt.close('all')
```



Examples with code #5



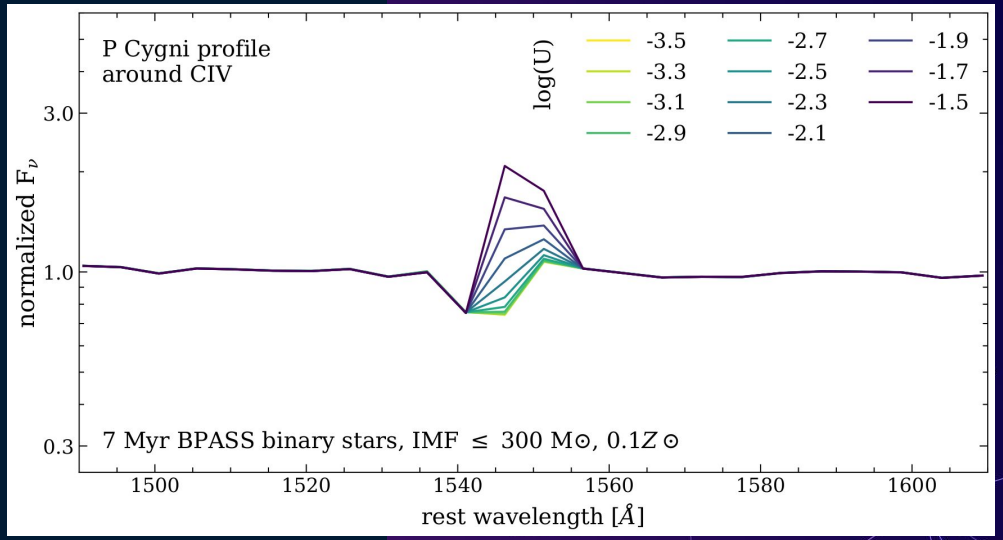
This is just some code I have to track gas prices in Texas, using the gas receipts I get every time I purchase gas.

The code is available in my GitHub repository related to this project.

Link to code:
github.com/aibhleog/gas-prices

Examples with code #6

```
'''
comment about what code does
'''

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from cloudy_func import * # written by TAH

__author__ = 'Taylor Hutchison'
__email__ = 'aibhleog@tamu.edu'

# ------------------------------------------ #
# -- running through series of the models -- #
# ------------------------------------------ #
u = np.arange(-3.5,-1.4,0.2)     # full range, 11 ionization points
zneb = 0.1                       # 0.1, 0.2, 0.3, 0.5 Zsolar
mass = 300                       # 300 or 100
stars = 'binary'                 # binary or single
age = [7,8]                      # 7, 7.477, or 8

fig = plt.figure(figsize=(8,8))
gs0 = gridspec.GridSpec(2,1,height_ratios=[1,1])

cmap = plt.get_cmap('viridis')
colors = [cmap(i) for i in np.linspace(1,0,len(u))] # makes a list of colors of a certain length
```



Examples with code #6

```
for a in range(2):
    ax = plt.subplot(gs0[a])
    df = get_cloudy_spec(f'{stars}_cont_{mass}',mass,age[a],zneb)

    for ion in u:
        i = u.tolist().index(ion) # getting the index to use for the color list
        plt.plot(df.wavelength,df.spectrum,color=colors[i],label=round(ion,1))

    # the 'transform' kwarg allows you to anchor the test to the dimensions of the plot
    # versus having to plug in x and y values that are relevant to what you're plotting
    # so you can do plt.text(0.5,0.5,'text',transform=plt.gca().transAxes) to have text
    # located directly in the middle of the plot
    text_kwargs = {'transform':plt.gca().transAxes,'fontsize':16} # easy way to set dictionary of kwargs
    plt.text(0.025,0.85,'P Cygni profile \naround CIV',**text_kwargs) # to use in many places
    plt.text(0.5,0.8,'log(U)',rotation=90,**text_kwargs) # also used here
    plt.text(0.025,0.05,'%s Myr BPASS %s stars, IMF $\leq$ %s M$\odot$, %s$ Z\odot$'\
             %(age[a],stars,mass,zneb),**text_kwargs) # and used here

    plt.legend(ncol=3) # three columns
    plt.xlim(1490,1610)
    plt.yscale('log')
    plt.ylim(0.25,6)
    plt.gca().set_yticks([0.3,1.,3,])
    plt.gca().set_yticklabels(['0.3','1.0','3.0',])

    plt.ylabel(r'normalized F$_{\nu}$',fontsize=17)
    plt.xlabel('rest wavelength [$\AA$]',fontsize=16,labelpad=5)

plt.tight_layout()
plt.show()
plt.close()
# ---------------------------------------- #
```