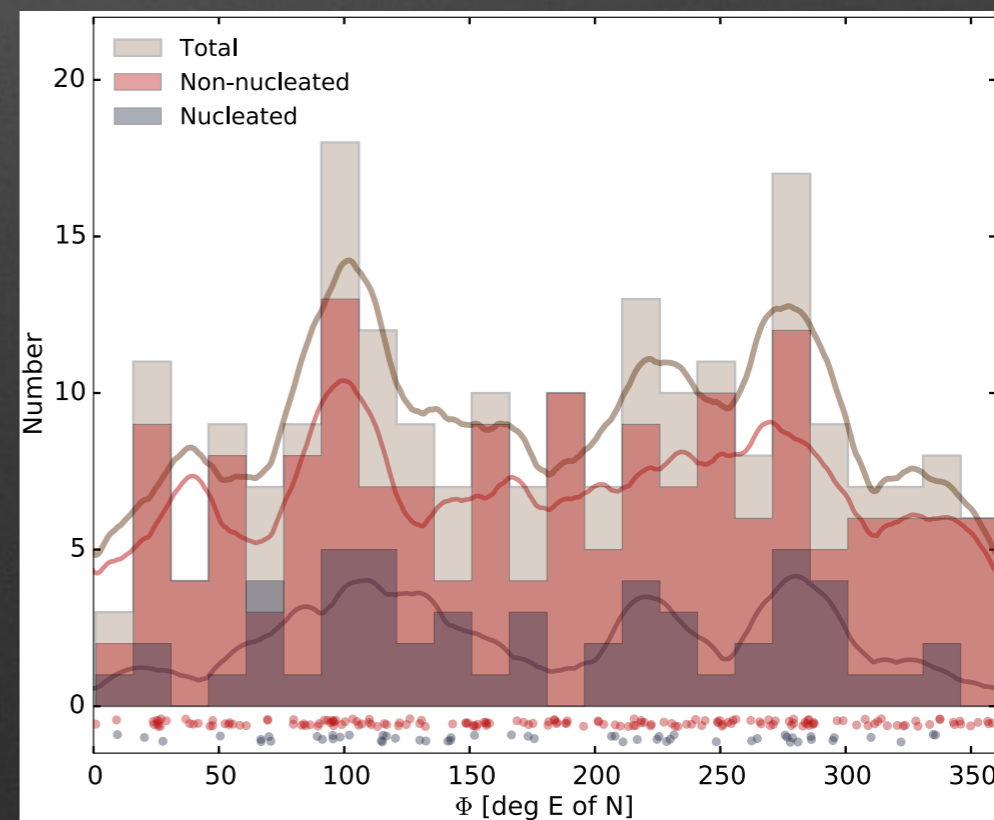
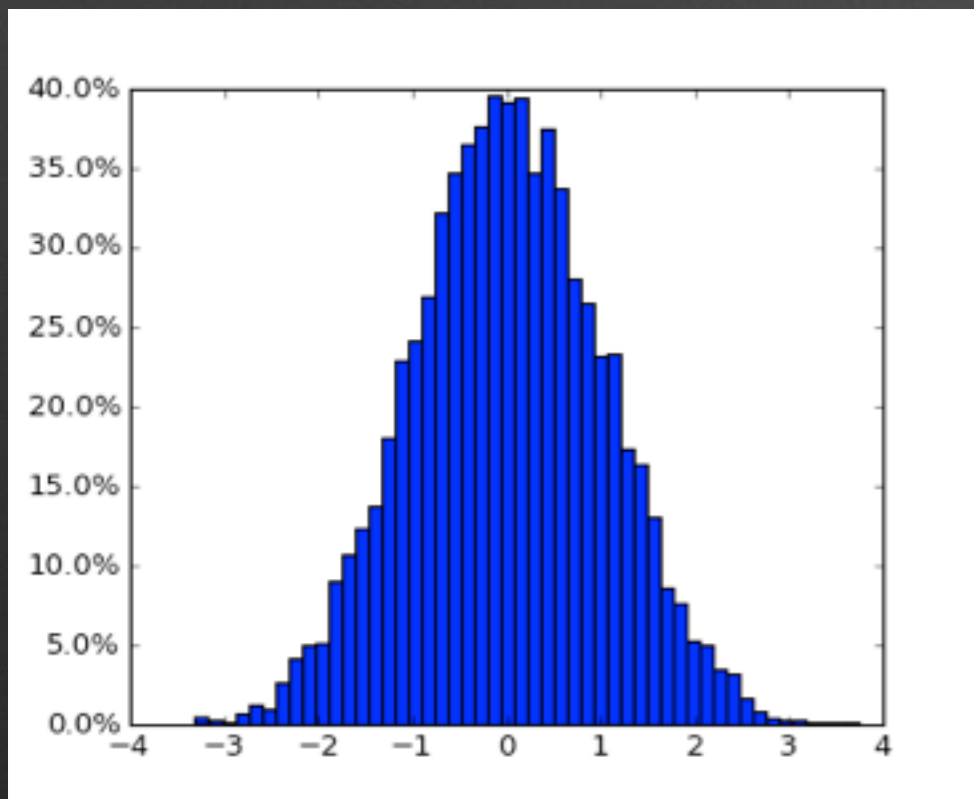
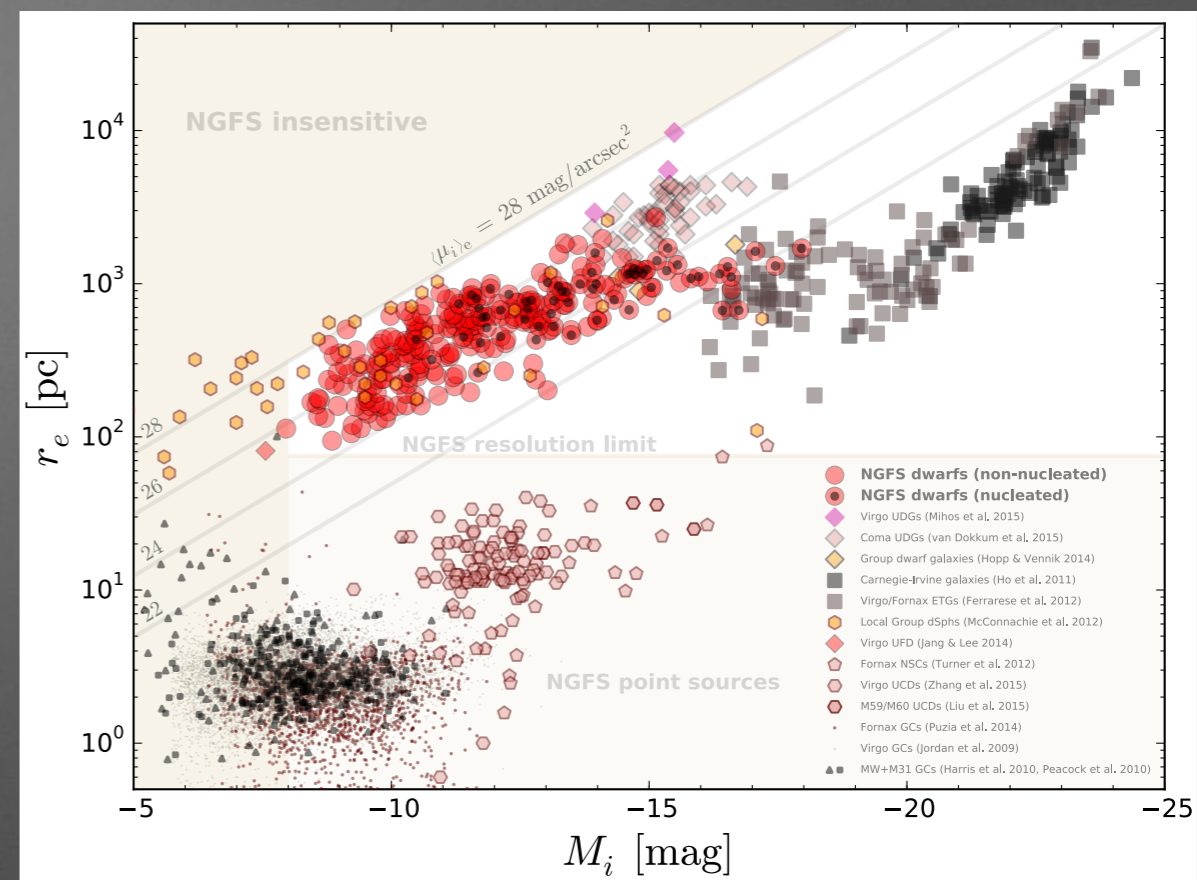
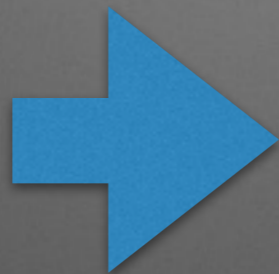
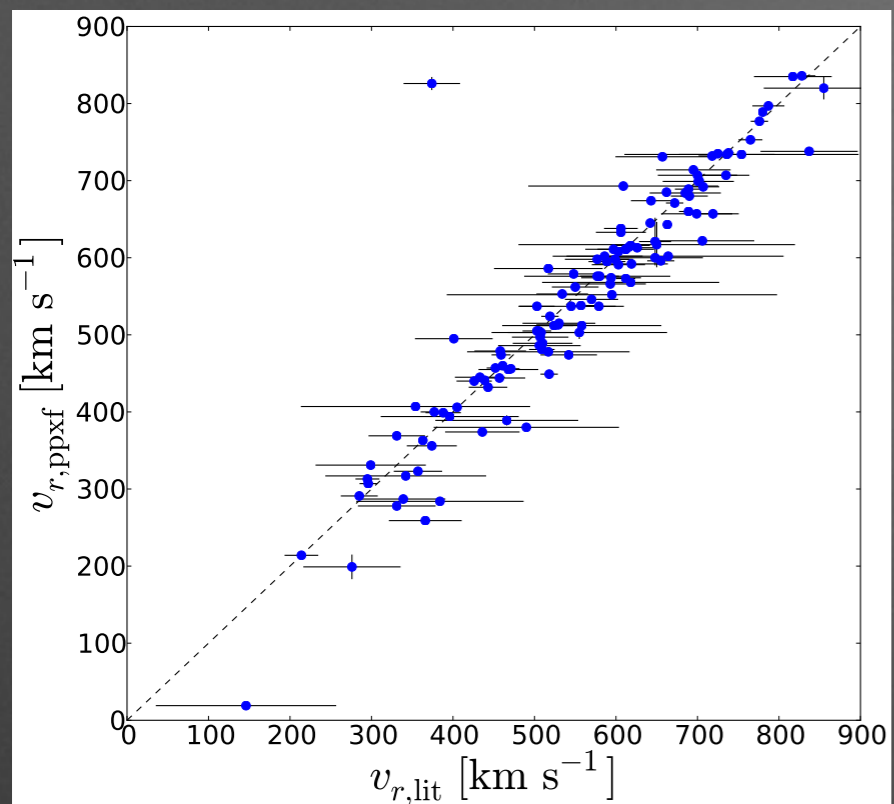


# Data Visualization with

(or, how to pimp your plots)

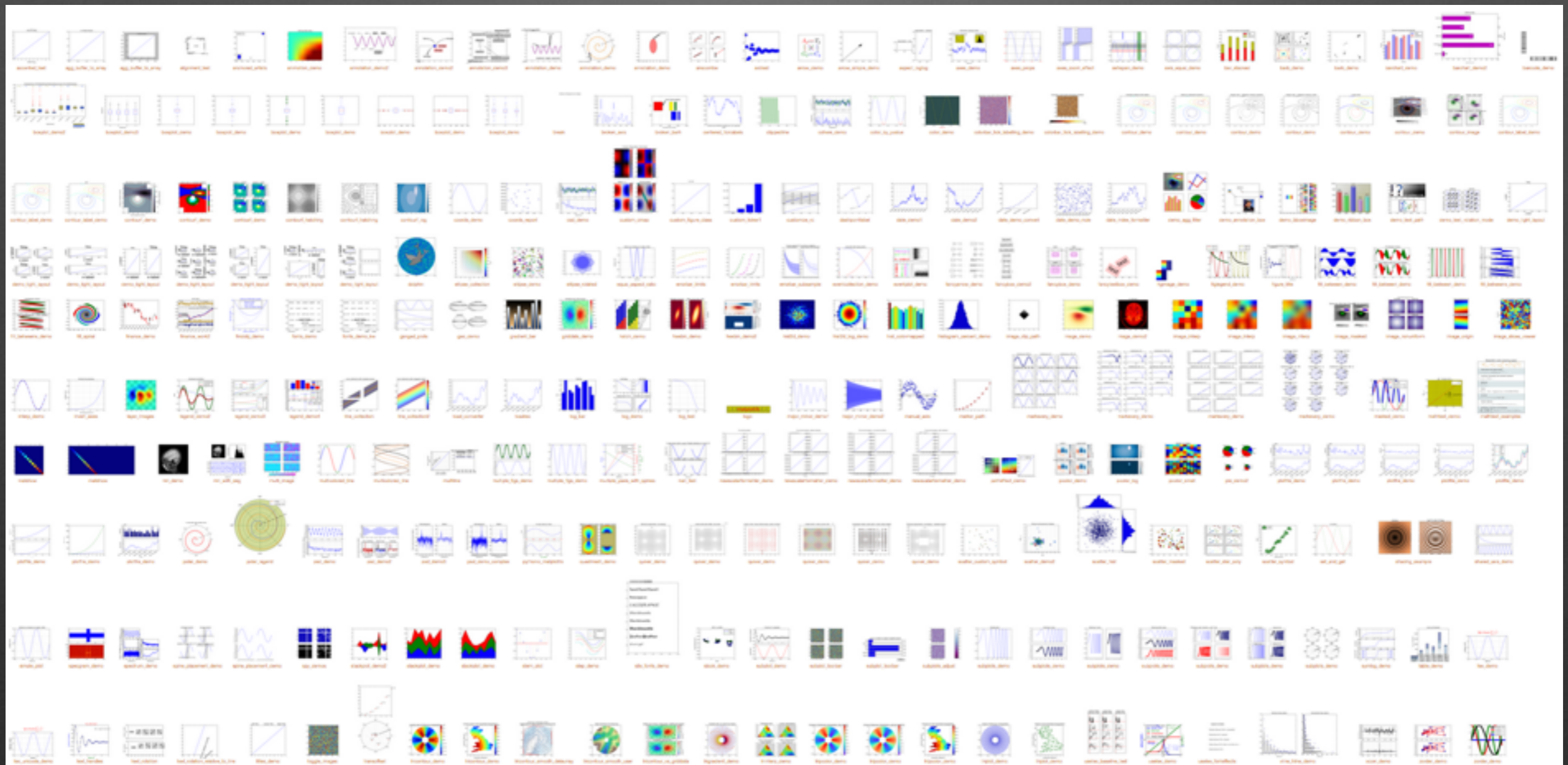


# What is



# ?

- 2D python plotting library for publication grade figures
- open source, [matplotlib.org](https://matplotlib.org)
- gallery with click-through examples for most plots you might need
- **FREE!**



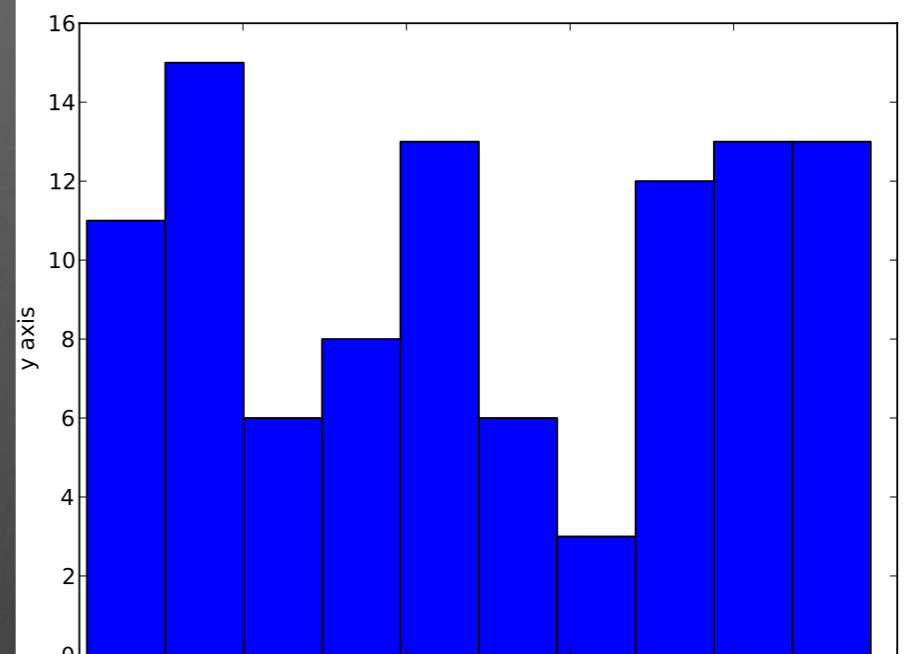
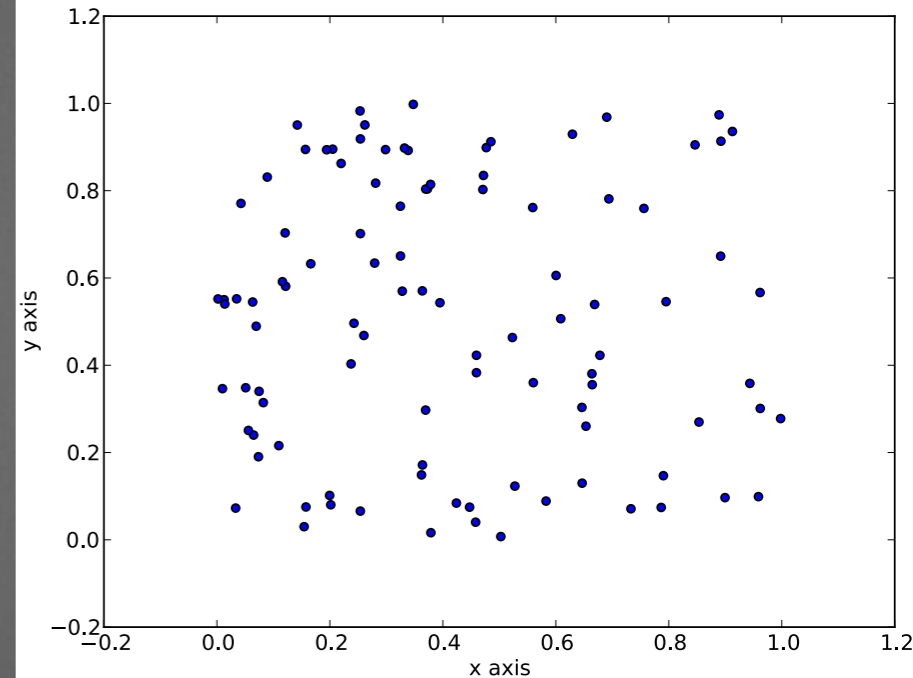
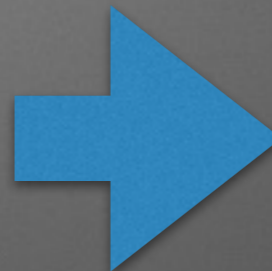
# A simple scatter plot and histogram:

- `np.loadtxt()` or `np.genfromtext()` to load data
- `plt.plot()` connects points with lines
- `plt.scatter()` plots only points
- `plt.hist()` creates histogram
- control number of bins with “bins=NN”

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

x = random.uniform(0,1,100)
y = random.uniform(0,1,100)

plt.figure()
plt.scatter(x,y)
plt.xlabel(r'x axis')
plt.ylabel(r'y axis')
plt.savefig('deleteme_scatter.pdf')
```



# Add a colour bar and legend:

- parameterize scatter data with “c=”
- add colour bar with `plt.colorbar()`
- add legend with `plt.legend()`

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

x = random.uniform(0,1,100)
y = random.uniform(0,1,100)

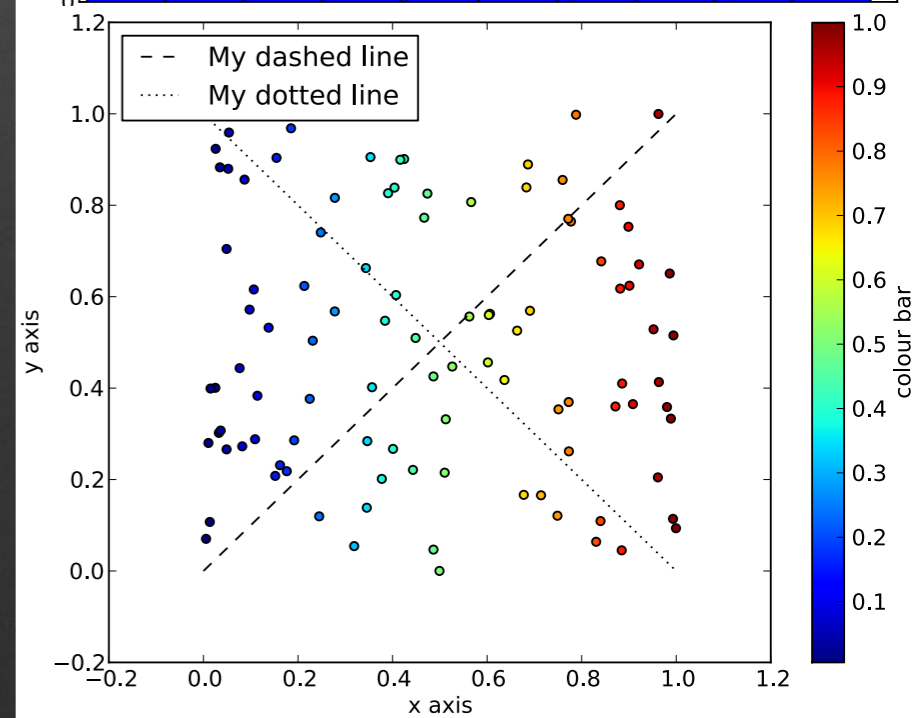
cm = plt.cm.get_cmap('jet')

plt.figure()
line1 = plt.plot([0,1],[0,1], 'k--')
line2 = plt.plot([0,1],[1,0], 'k:')
plt.scatter(x,y,c=x,cmap=cm)
plt.xlabel(r'x axis')
plt.ylabel(r'y axis')

cb = plt.colorbar()
cb.set_label(r'colour bar')

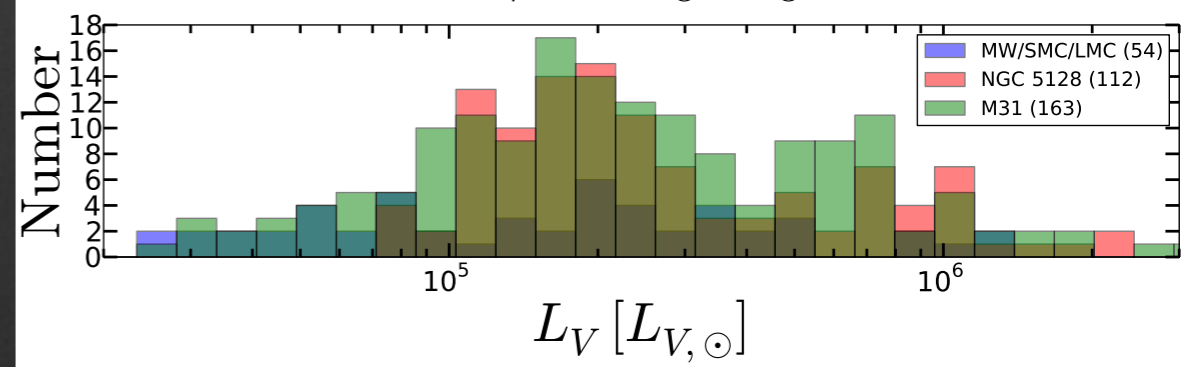
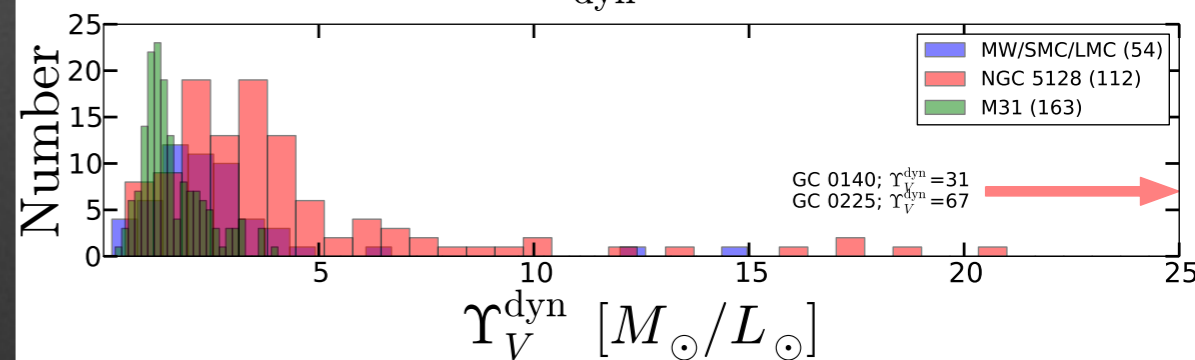
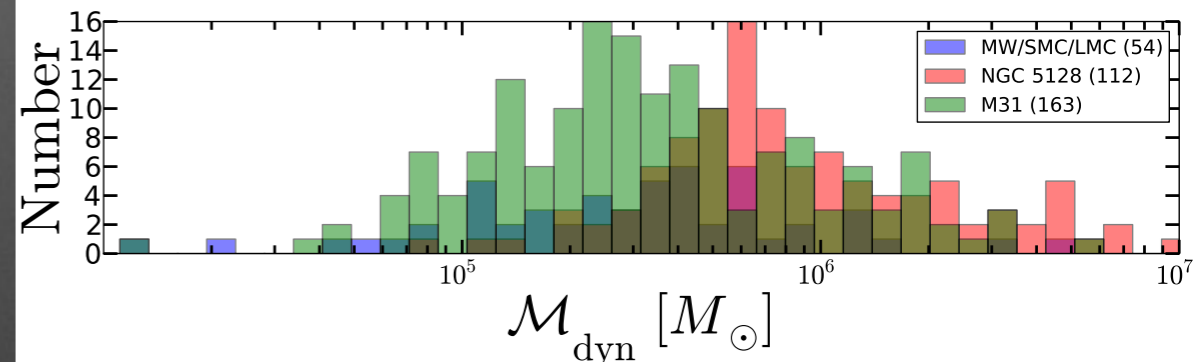
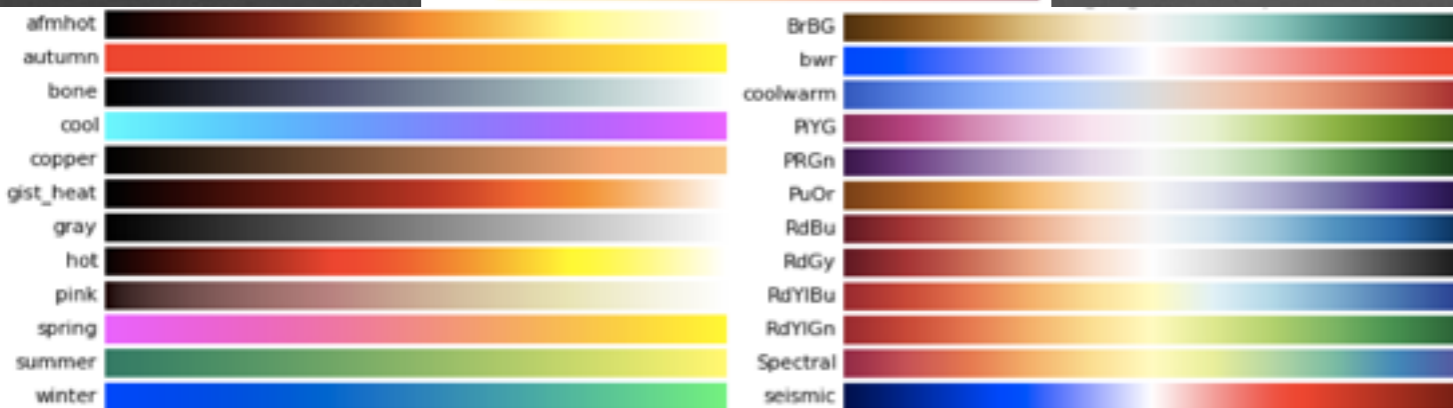
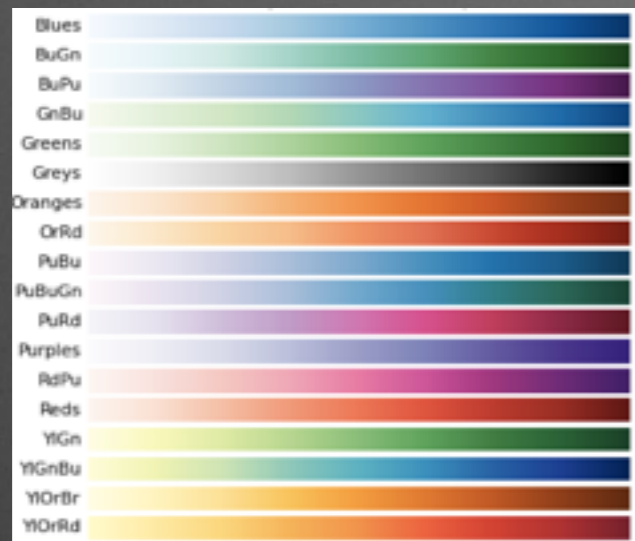
plt.legend((line1,line2),(r'My dashed line',r'My dotted line'),loc=2)

plt.savefig('deleteme_colourleg.pdf')
```



# Colours:

- large library of named colours; easy
- 16 million colours via html hex strings; pretty
- html hex strings, e.g. “eeeeff”



# “Alpha” channels:

- add opacity for crowded figures
- “alpha=N.N”
  - 0.0 -> fully transparent
  - 1.0 -> fully opaque

# Inset figures:

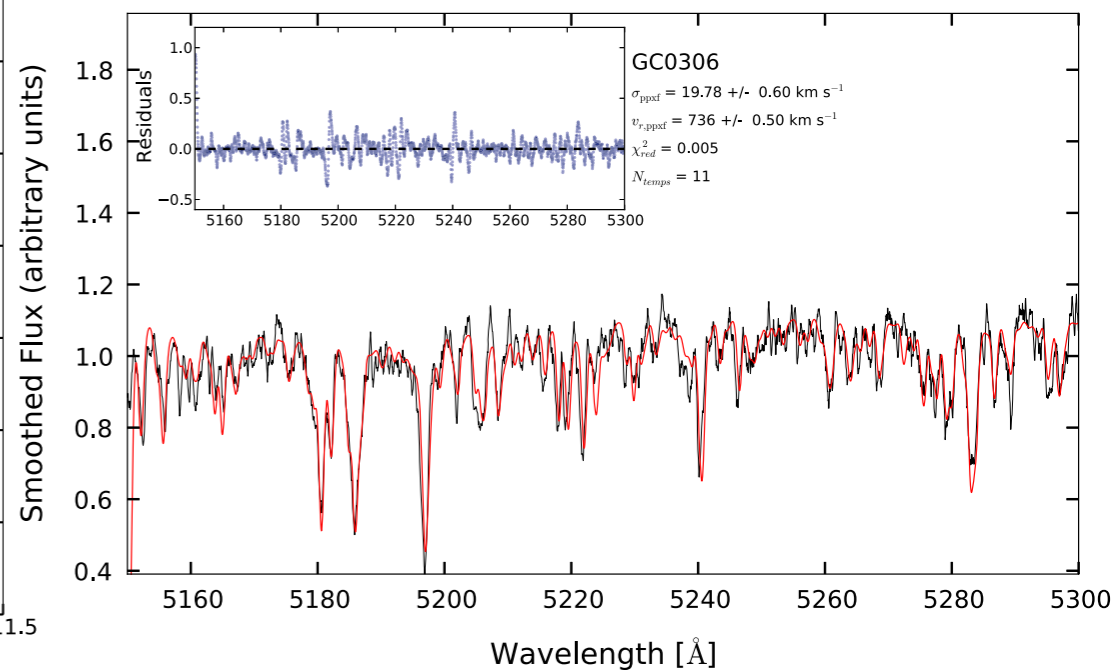
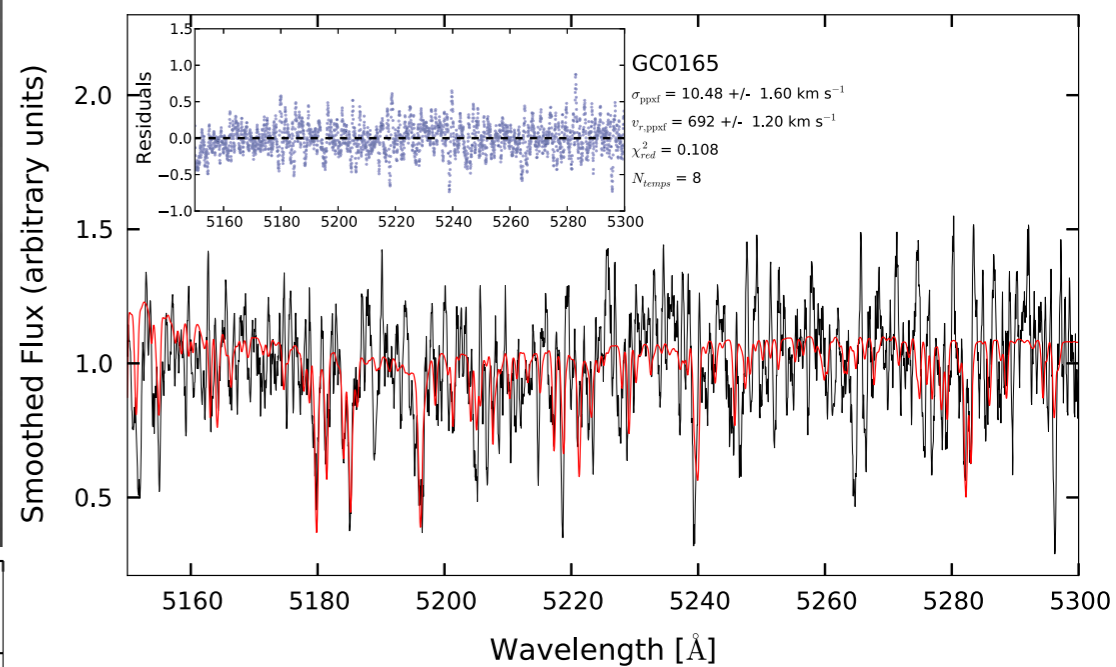
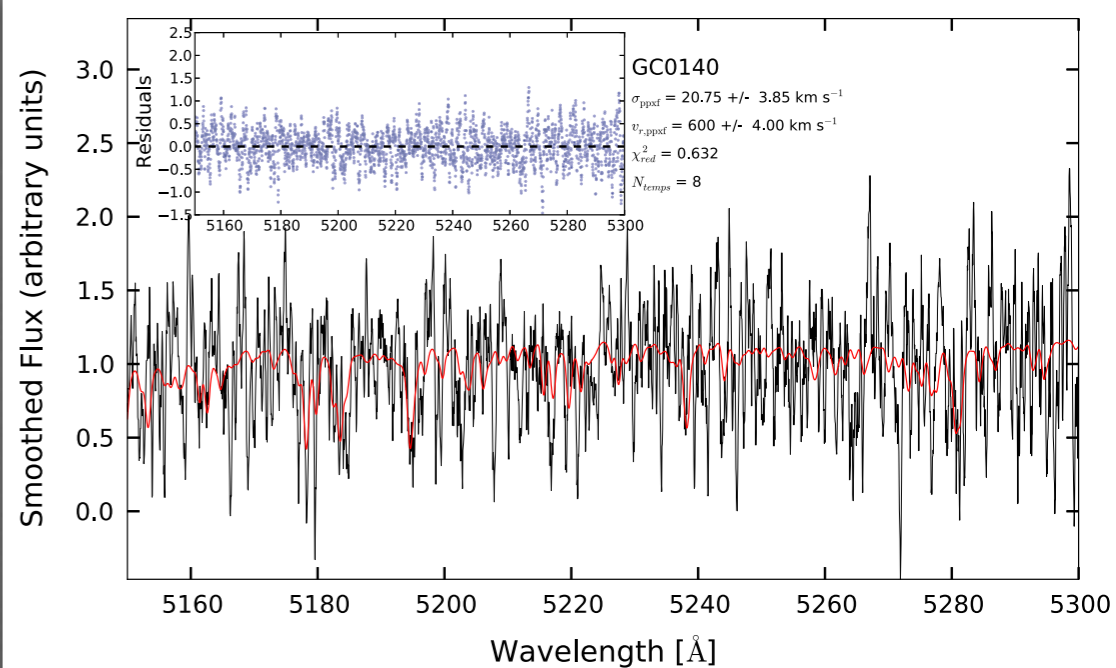
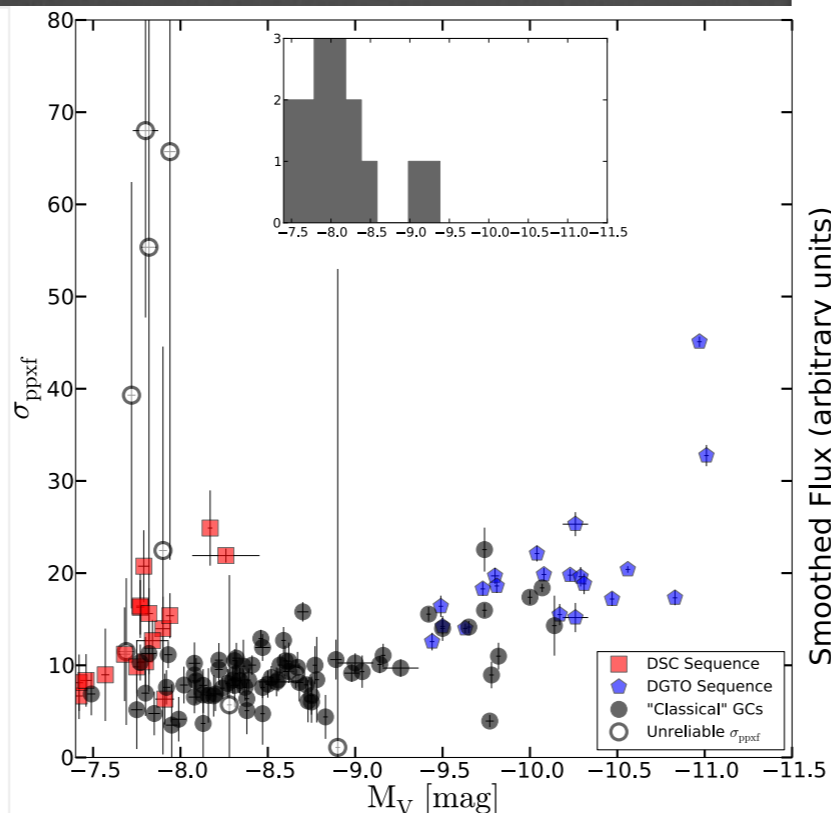
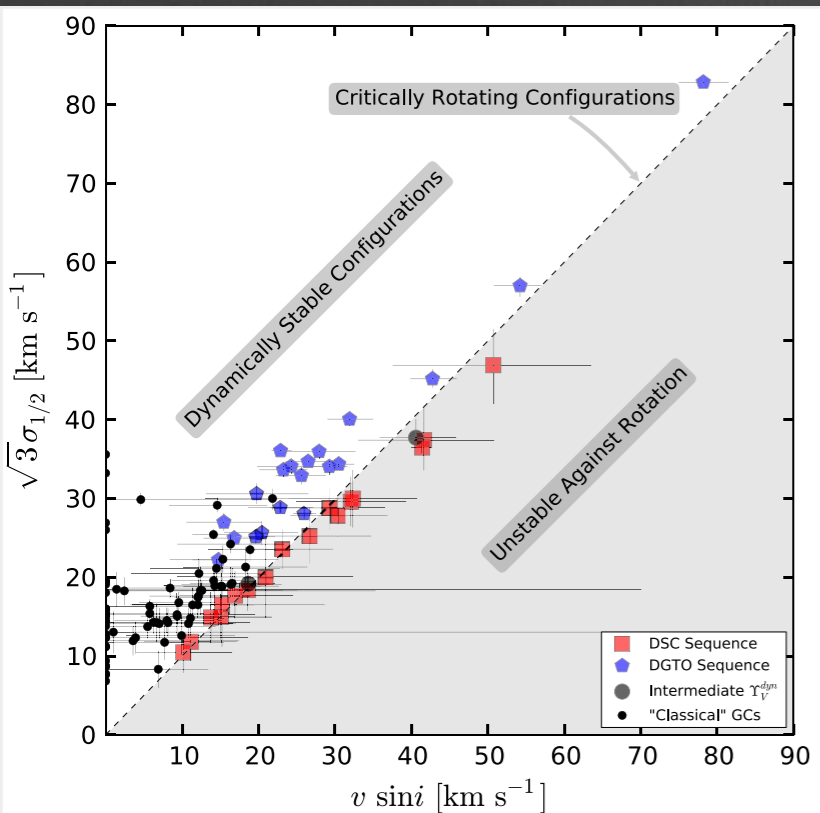
- `plt.axes([x0, y0, Δx, Δy])` to create new canvas

# Multi-panel figures

- `plt.subplot(n_row, n_col, N)` creates  $n_{\text{row}} \times n_{\text{col}}$  panel figure, where N is current canvas

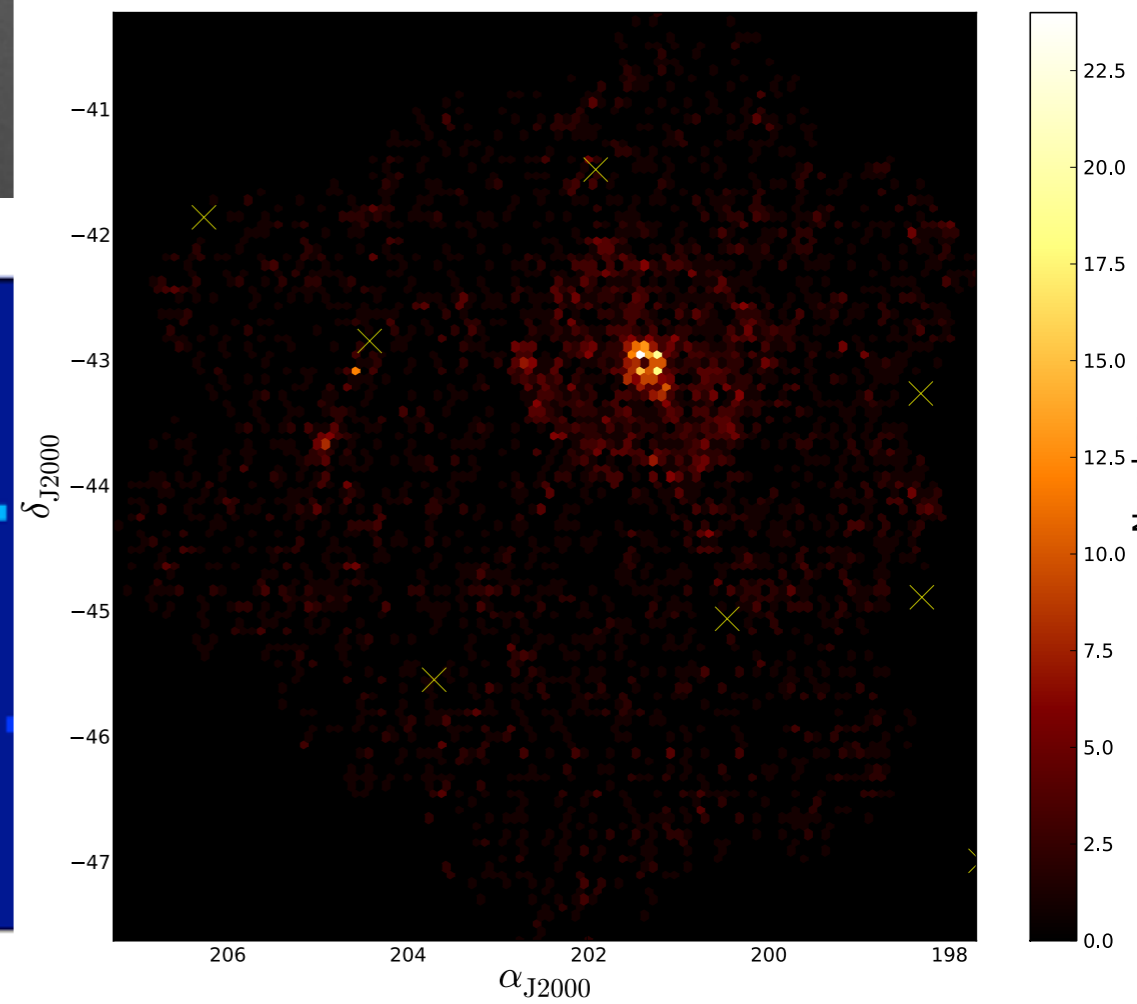
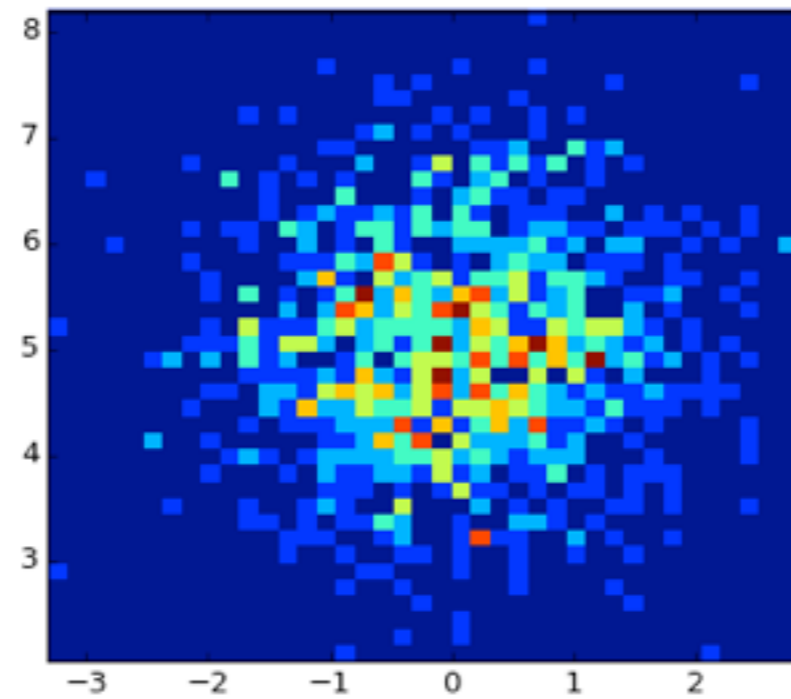
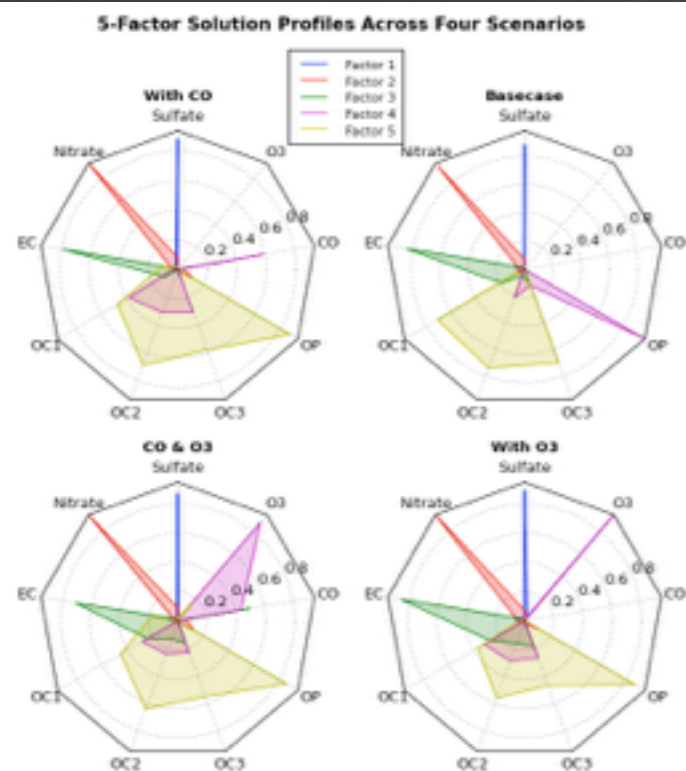
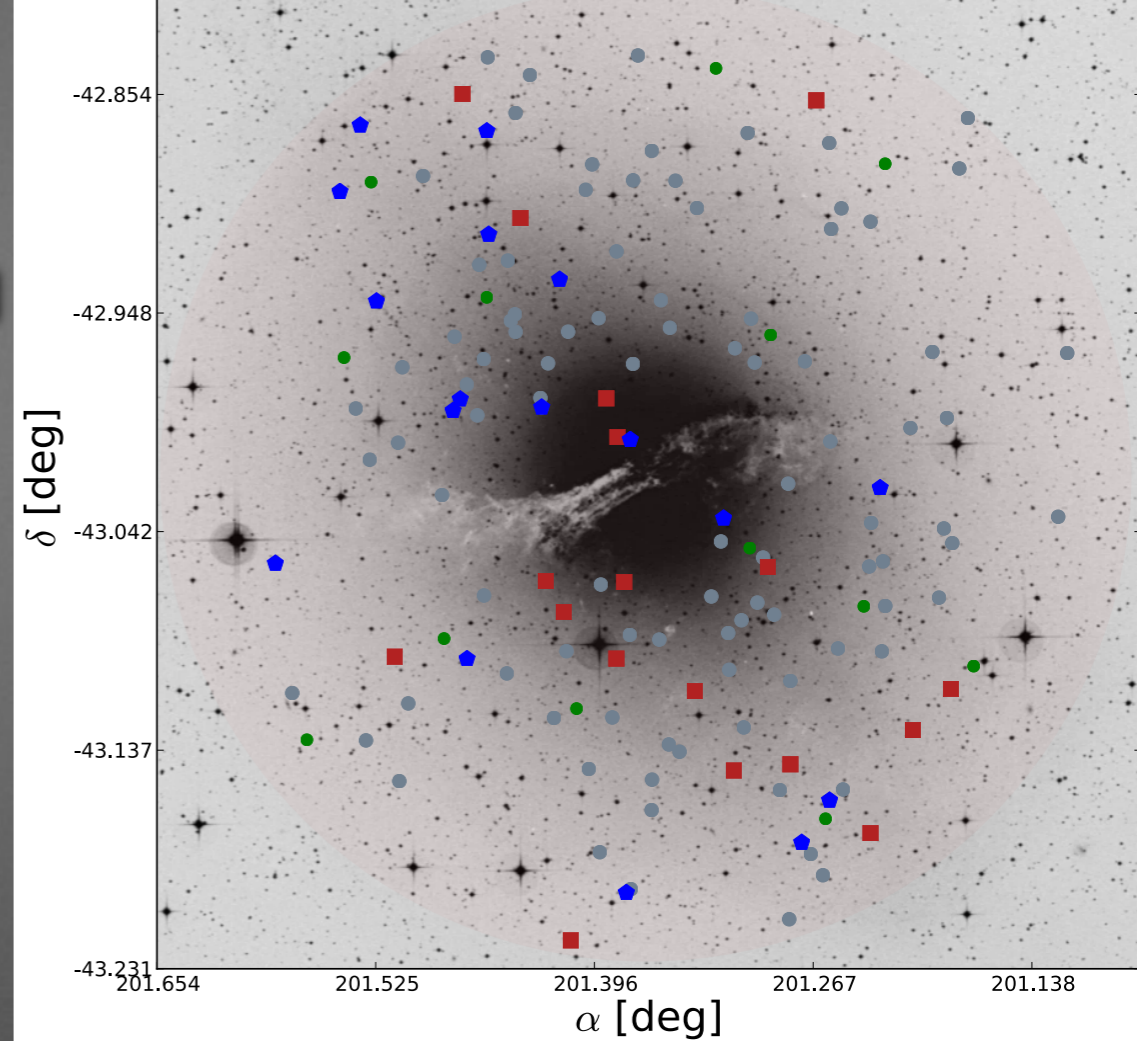
# Annotations:

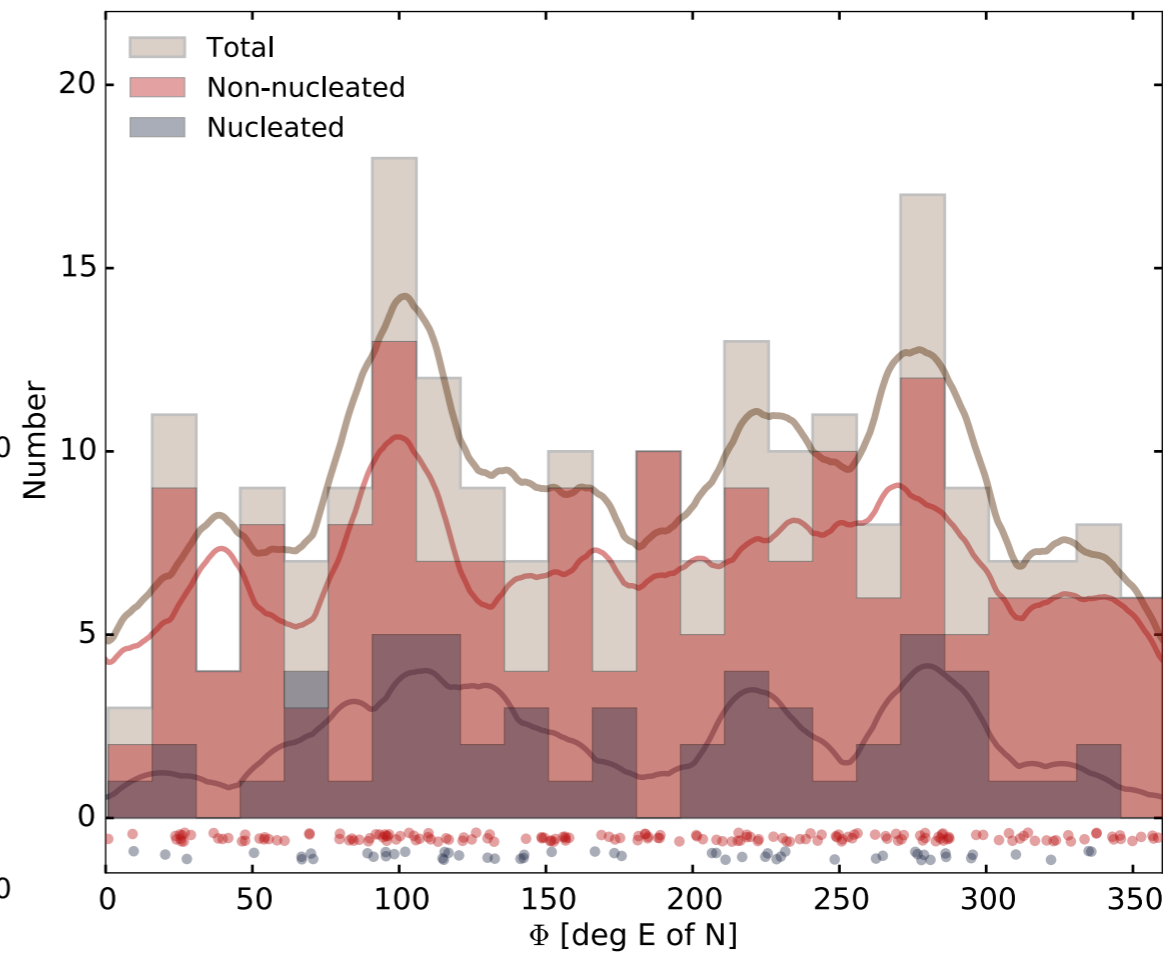
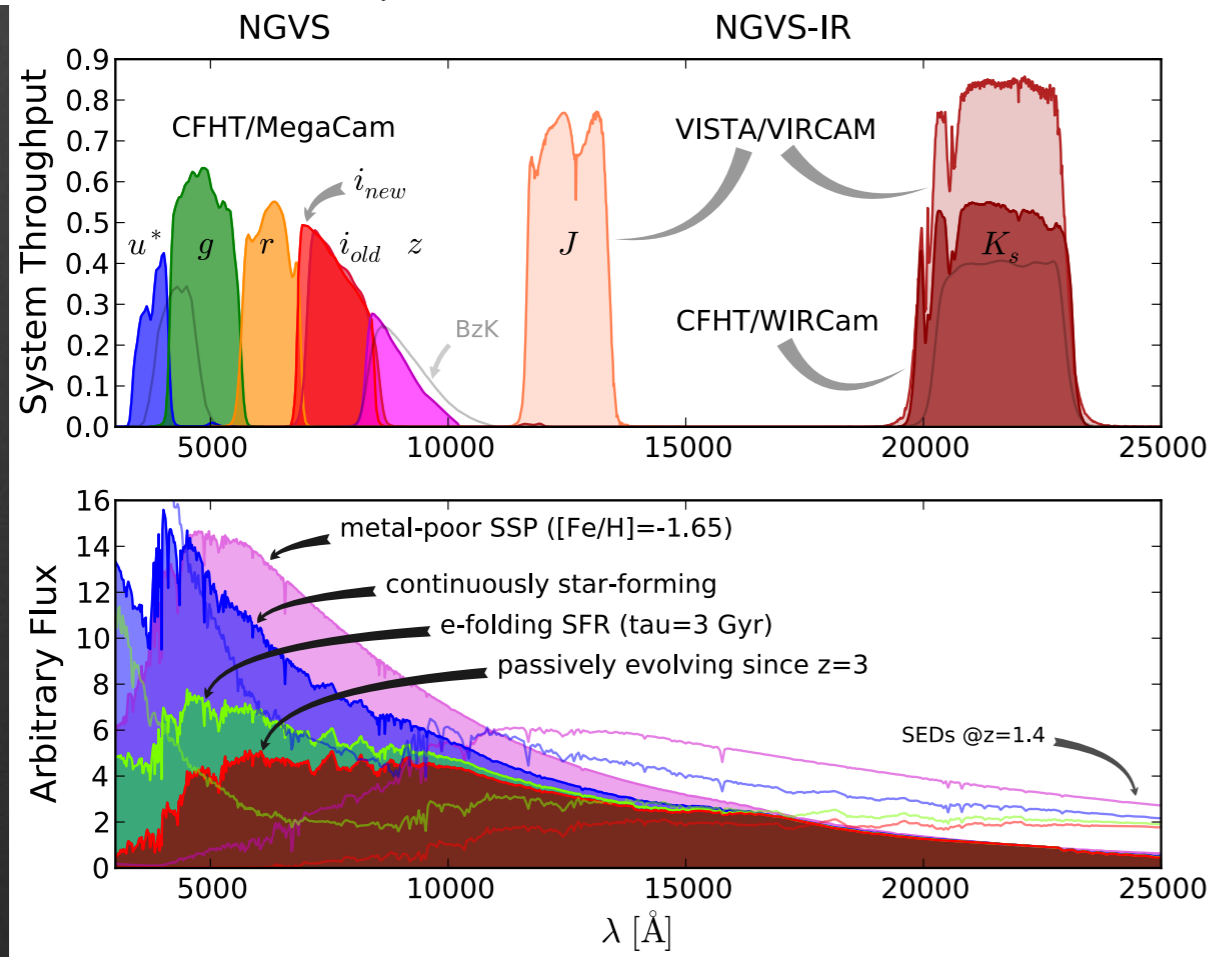
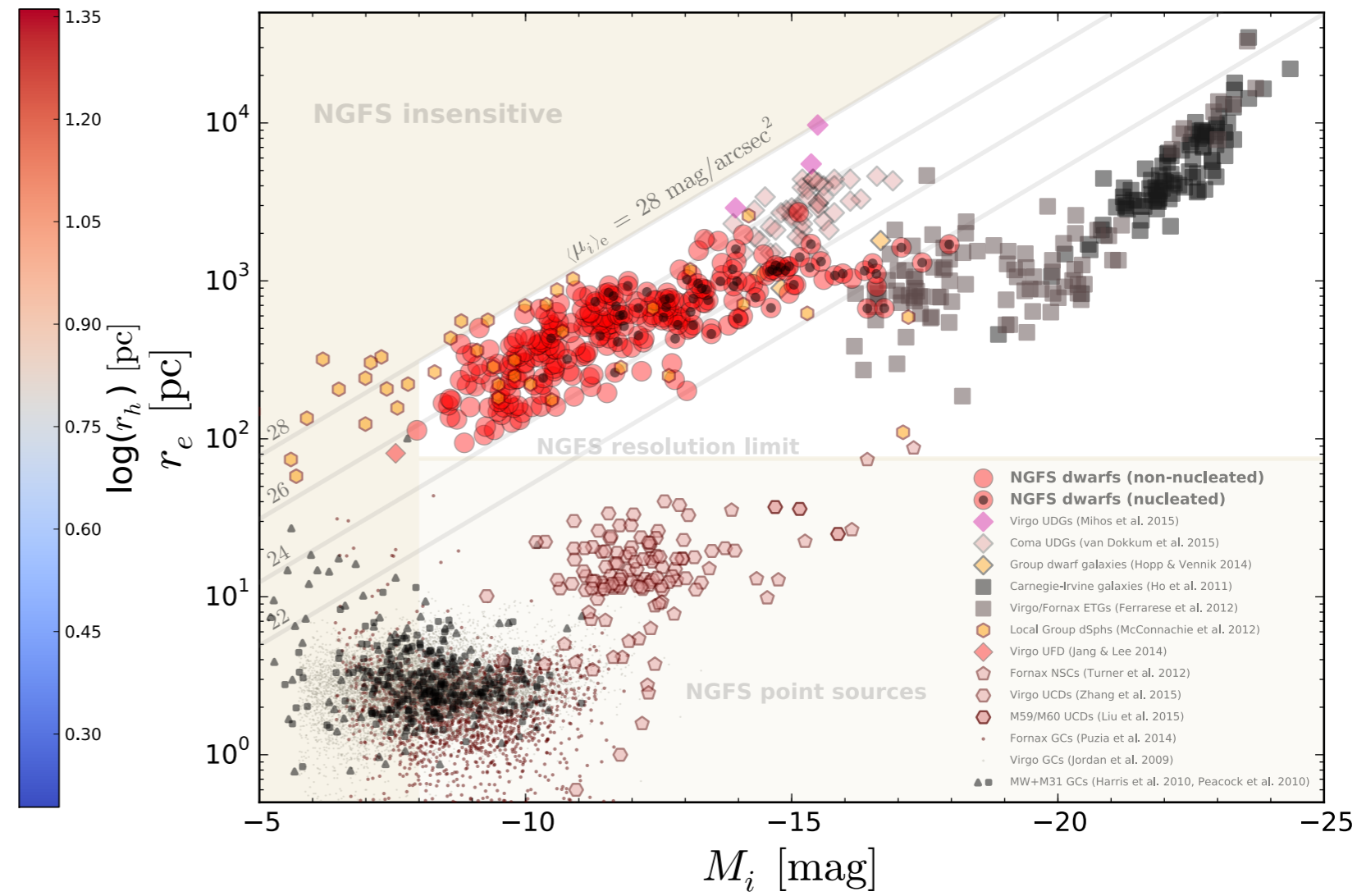
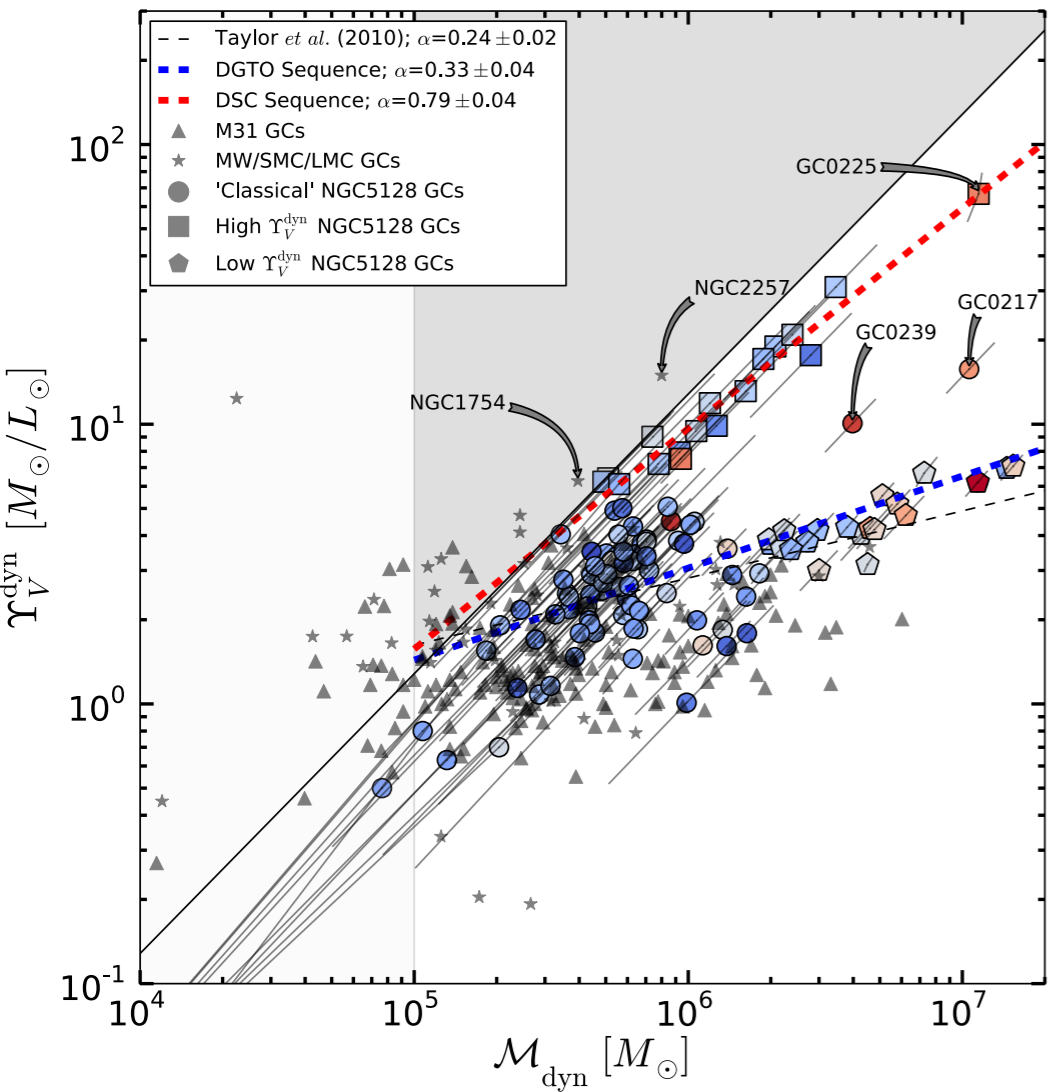
- `plt.annotate(string, xy=(x0, y0))`
- many options for angles, arrows, etc.




# Special Plots:


- `plt.imshow()`
  - display image (e.g. jpg) which can be plotted over
- `plt.2dhist()`
  - 2D histogram
- `plt.hexbin()`
  - 2D histogram with hexagonal bins (smoother look)
- `matplotlib.patches`
  - library to load various shapes
  - create “radar” charts





# Summary:

 **matplotlib** is a powerful, easy to use/learn python 2D plotting package

- loads of online support e.g. [matplotlib.org](http://matplotlib.org), [stackoverflow.com](http://stackoverflow.com), etc.
- days of “simple” figures are coming to an end
  - page charges, colour figures, etc. put premium on effective/efficient data visualization
-  **matplotlib** tricks enable great information density on single plot, without loss of readability
- colour schemes, opacity, figure insets, etc. make it easy to create eye catching figures for papers and posters
  - easy to learn and implement