

Coding tip #1: Boolean flags to define data subsets

imagine you have magnitudes/colors for stars and you want to make a color-magnitude diagram and then select a sample of stars that define some color and magnitude criteria.

(Note: in the Gaia problems on the HW set, we use the Gaia G magnitude ('phot_g_mean_mag') and Bp-Rp color ('bp_rp'), while this example uses SDSS g mags and g-r color, but the coding concepts are the same.....)

```
g,g_r = data['g'], data['g'] - data['r'] # get mag and color out of dataset
plt.scatter(g_r,g) # make a CMD (remember to set your axes intelligently)
```

```
# now lets say we want things in the magnitude range g = 8 to 10 and color range g-r = 1 to 1.5
```

```
want_mag = np.logical_and(g>=8, g<=10)
want_color = np.logical_and(g_r >=1, g_r<=1.5)
want = np.logical_and(want_mag,want_color)
```

```
# and lets say that we also want things to be main sequence stars (using the codes in HW #2, problem #4):
```

```
want = np.logical_and(want, data['star_type'] == 1) # you can keep stacking 'wants' like this....
```

```
# and then make sure we selected right by overplotting the sample on the cmd
```

```
plt.scatter(g_r[want],g[want],color='red') # do the red points fall where they should?
```

Coding tip #2: Boolean flags to fit data

Imagine you have a set of x,y points and you want to fit a line to them, but not over the whole dataset.

```
# First plot your data
```

```
plt.scatter(x,y)
```

```
# then decide which points you want to fit, say only data in a certain range of x.
```

```
xfit_min, xfit_max = 5, 15
```

```
fit_range = np.logical_and(x>xfit_min, x<xfit_max)
```

```
coeff,cov=np.polyfit(x[fit_range],y[fit_range],1,cov=True)
```

```
# see python help page link on ASTR 323 website for a fitting example that prints out
```

```
# fit parameters and uncertainties.
```

```
# then always overplot your fit !
```

```
xfit=np.linspace(xfit_min, xfit_max,100)
```

```
polynomial=np.poly1d(coeff)
```

```
plt.plot(xfit,polynomial(xfit),color='green',lw=3)
```

```
# and always plot the points that weren't used in the fit!
```

```
plt.scatter(x[~fit_range],y[~fit_range])
```