Day 6, Part 1 - Data processing, Info Viz

While we'll focus a lot of scientific viz of your sims, we can also take this opportunity to do some "info viz" which can provide context to your sci viz. Info viz is an important skill to know how to do as well!

Let's take a step back and look at some larger planet data to see how we might process a large list of data. In this example, we'll use more Kepler data.

While filtering is important only sometimes for simulated data, it is generally necessary for observational data & if we want to compare our simulations to observations.

We'll use the "pandas" package to do this which can be a useful thing to know how to use anyway!

In [1]:
```
# import our usual stuffs
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
# now, import pandas
import pandas as pd
```

In [3]:
```
# now let's read in the kepler confirmed planets dataset
planets = pd.read_csv('https://jnaiman.github.io/csci-p-14110/lesson06/d
ata/planets_2019.07.12_17.16.25.csv',
                     sep=",", comment="#")
#note: feel free to download this and read from your download as well
```

In [4]:
```
planets
# formatting here is sort of nice
```

Out[4]:

	pl_hostname	pl_letter	pl_name	pl_discmethod	pl_controvflag	pl_pnum
0	11 Com	b	11 Com b	Radial Velocity	0	1
1	11 UMi	b	11 UMi b	Radial Velocity	0	1
2	14 And	b	14 And b	Radial Velocity	0	1
3	14 Her	b	14 Her b	Radial Velocity	0	1
4	16 Cyg B	b	16 Cyg B b	Radial Velocity	0	1
5	18 Del	b	18 Del b	Radial Velocity	0	1
6	1RXS J160929.1-210524	b	1RXS J160929.1-210524 b	Imaging	0	1
7	24 Boo	b	24 Boo b	Radial Velocity	0	1
8	24 Sex	b	24 Sex b	Radial Velocity	0	2
9	24 Sex	c	24 Sex c	Radial Velocity	0	2
10	2MASS J01225093-2439505	b	2MASS J01225093-2439505 b	Imaging	0	1
11	2MASS J02192210-3925225	b	2MASS J02192210-3925225 b	Imaging	0	1
12	2MASS J04414489+2301513	b	2MASS J04414489+2301513 b	Imaging	0	1
13	2MASS J12073346-3932539	b	2MASS J12073346-3932539 b	Imaging	0	1
14	2MASS J19383260+4603591	b	2MASS J19383260+4603591 b	Eclipse Timing Variations	0	1
15	2MASS J21402931+1625183 A	b	2MASS J21402931+1625183 A b	Imaging	0	1
16	2MASS J22362452+4751425	b	2MASS J22362452+4751425 b	Imaging	0	1
17	30 Ari B	b	30 Ari B b	Radial Velocity	0	1
18	4 UMa	b	4 UMa b	Radial Velocity	0	1
19	42 Dra	b	42 Dra b	Radial Velocity	0	1
20	47 UMa	b	47 UMa b	Radial Velocity	0	3
21	47 UMa	c	47 UMa c	Radial Velocity	0	3
22	47 UMa	d	47 UMa d	Radial Velocity	0	3
23	51 Eri	b	51 Eri b	Imaging	0	1
24	51 Peg	b	51 Peg b	Radial Velocity	0	1
25	55 Cnc	b	55 Cnc b	Radial Velocity	0	5
26	55 Cnc	c	55 Cnc c	Radial Velocity	0	5
27	55 Cnc	d	55 Cnc d	Radial Velocity	0	5

	pl_hostname	pl_letter	pl_name	pl_discmethod	pl_controvflag	pl_pnum
28	55 Cnc	e	55 Cnc e	Radial Velocity	0	5
29	55 Cnc	f	55 Cnc f	Radial Velocity	0	5
...	...	...	...	...	...	...
3986	eps CrB	b	eps CrB b	Radial Velocity	0	1
3987	eps Eri	b	eps Eri b	Radial Velocity	0	1
3988	eps Tau	b	eps Tau b	Radial Velocity	0	1
3989	gam 1 Leo	b	gam 1 Leo b	Radial Velocity	0	1
3990	gam Cep	b	gam Cep b	Radial Velocity	0	1
3991	gam Lib	b	gam Lib b	Radial Velocity	0	2
3992	gam Lib	c	gam Lib c	Radial Velocity	0	2
3993	iot Dra	b	iot Dra b	Radial Velocity	0	1
3994	kap And	b	kap And b	Imaging	0	1
3995	kap CrB	b	kap CrB b	Radial Velocity	0	1
3996	mu Leo	b	mu Leo b	Radial Velocity	0	1
3997	nu Oph	b	nu Oph b	Radial Velocity	0	2
3998	nu Oph	c	nu Oph c	Radial Velocity	0	2
3999	ome Ser	b	ome Ser b	Radial Velocity	0	1
4000	omi CrB	b	omi CrB b	Radial Velocity	0	1
4001	omi UMa	b	omi UMa b	Radial Velocity	0	1
4002	HD 39091	c	pi Men c	Transit	0	2
4003	psi 1 Dra B	b	psi 1 Dra B b	Radial Velocity	0	1
4004	rho CrB	b	rho CrB b	Radial Velocity	1	2
4005	rho CrB	c	rho CrB c	Radial Velocity	0	2
4006	tau Boo	b	tau Boo b	Radial Velocity	0	1
4007	tau Cet	e	tau Cet e	Radial Velocity	0	4
4008	tau Cet	f	tau Cet f	Radial Velocity	0	4
4009	tau Cet	g	tau Cet g	Radial Velocity	0	4
4010	tau Cet	h	tau Cet h	Radial Velocity	0	4
4011	tau Gem	b	tau Gem b	Radial Velocity	0	1
4012	ups And	b	ups And b	Radial Velocity	0	3
4013	ups And	c	ups And c	Radial Velocity	0	3
4014	ups And	d	ups And d	Radial Velocity	0	3
4015	xi Aql	b	xi Aql b	Radial Velocity	0	1

4016 rows × 88 columns

In [5]:
```
# skip this
# how many entries are there? as an iterable
#planets.index
```

In [6]:
```
planets.loc[0:3] #easy to grab subsets - here by label
#planets.loc? #easy to grab subsets - here by label
```

Out[6]:

	pl_hostname	pl_letter	pl_name	pl_discmethod	pl_controvflag	pl_pnum	pl_orbper	pl_orbpe
0	11 Com	b	11 Com b	Radial Velocity	0	1	326.03000	
1	11 UMi	b	11 UMi b	Radial Velocity	0	1	516.21997	
2	14 And	b	14 And b	Radial Velocity	0	1	185.84000	
3	14 Her	b	14 Her b	Radial Velocity	0	1	1773.40002	

4 rows × 88 columns

In [7]:
```
planets.columns
# names of columns
```

Out[7]:
```
Index(['pl_hostname', 'pl_letter', 'pl_name', 'pl_discmethod',
       'pl_controvflag', 'pl_pnum', 'pl_orbper', 'pl_orbpererr1',
       'pl_orbpererr2', 'pl_orbperlim', 'pl_orbsmax', 'pl_orbsmaxerr1',
       'pl_orbsmaxerr2', 'pl_orbsmaxlim', 'pl_orbeccen', 'pl_orbeccener
r1',
       'pl_orbeccenerr2', 'pl_orbeccenlim', 'pl_orbincl', 'pl_orbincler
r1',
       'pl_orbinclerr2', 'pl_orbincllim', 'pl_bmassj', 'pl_bmassjerr1',
       'pl_bmassjerr2', 'pl_bmassjlim', 'pl_bmassprov', 'pl_radj',
       'pl_radjerr1', 'pl_radjerr2', 'pl_radjlim', 'pl_dens', 'pl_dense
rr1',
       'pl_denserr2', 'pl_denslim', 'ra_str', 'ra', 'dec_str', 'dec',
       'st_dist', 'st_disterr1', 'st_disterr2', 'st_distlim', 'gaia_dis
t',
       'gaia_disterr1', 'gaia_disterr2', 'gaia_distlim', 'st_optmag',
       'st_optmagerr', 'st_optmaglim', 'st_optband', 'gaia_gmag',
       'gaia_gmagerr', 'gaia_gmaglim', 'st_teff', 'st_tefferr1', 'st_te
fferr2',
       'st_tefflim', 'st_mass', 'st_masserr1', 'st_masserr2', 'st_massl
im',
       'st_rad', 'st_raderr1', 'st_raderr2', 'st_radlim', 'pl_massj',
       'pl_massjerr1', 'pl_massjerr2', 'pl_massjlim', 'pl_rade', 'pl_ra
deerr1',
       'pl_radeerr2', 'pl_radelim', 'pl_disc', 'pl_mnum', 'st_sp', 'st_
spstr',
       'st_sperr', 'st_splim', 'st_lum', 'st_lumerr1', 'st_lumerr2',
       'st_lumlim', 'st_age', 'st_ageerr1', 'st_ageerr2', 'st_agelim'],
      dtype='object')
```

In [6]:
```
# skip
#planets.loc[0:10]["pl_orbeccen"] # grab 1-10 entries, and print out the
eccentricites of those entries
# notice there are some NaN's -> these just don't have entries
```

In [9]:
```
# what are the names of the unique host stars?
planets["pl_hostname"].unique()
```

Out[9]:
```
array(['11 Com', '11 UMi', '14 And', ..., 'tau Gem', 'ups And', 'xi Aq
l'],
      dtype=object)
```

In [10]:
```
planets["pl_hostname"].nunique() # how many unique host stars?
```

Out[10]:
```
2994
```

In [11]:
```
# if you are used to R at all, this is sort of like "summary" function,
 but basically giving some
# summary statistics for the numerical data in our dataset
planets.describe()
# note that while things like the statistics for the orbital period are
 interesting
# the "mean" of the pl_controvflag which is a flag if this planet is con
troversal or not
#  is essentially meaningless
```

Out[11]:

	pl_controvflag	pl_pnum	pl_orbper	pl_orbpererr1	pl_orbpererr2	pl_orbperlim	pl_
count	4016.000000	4016.000000	3.907000e+03	3.775000e+03	3.775000e+03	3941.000000	233
mean	0.002739	1.772410	2.326363e+03	1.051761e+03	-1.082432e+03	-0.000507	
std	0.052271	1.159506	1.171632e+05	5.948289e+04	5.968316e+04	0.039020	8
min	0.000000	1.000000	9.070629e-02	0.000000e+00	-3.650000e+06	-1.000000	
25%	0.000000	1.000000	4.516936e+00	1.600000e-05	-1.165000e-03	0.000000	
50%	0.000000	1.000000	1.193212e+01	9.500000e-05	-9.600000e-05	0.000000	
75%	0.000000	2.000000	4.231980e+01	1.173500e-03	-1.600000e-05	0.000000	
max	1.000000	8.000000	7.300000e+06	3.650000e+06	0.000000e+00	1.000000	250

8 rows × 79 columns

In [7]:
```
# skip
# we can also search for subsets easily
# we can look for only circular orbits
# -> look for eccentricity == 0
#planets.loc[planets["pl_orbeccen"] == 0.0]
```

In [8]:
```
# skip
# or very eccentric orbits
#planets.loc[planets["pl_orbeccen"] >= 0.9]
```

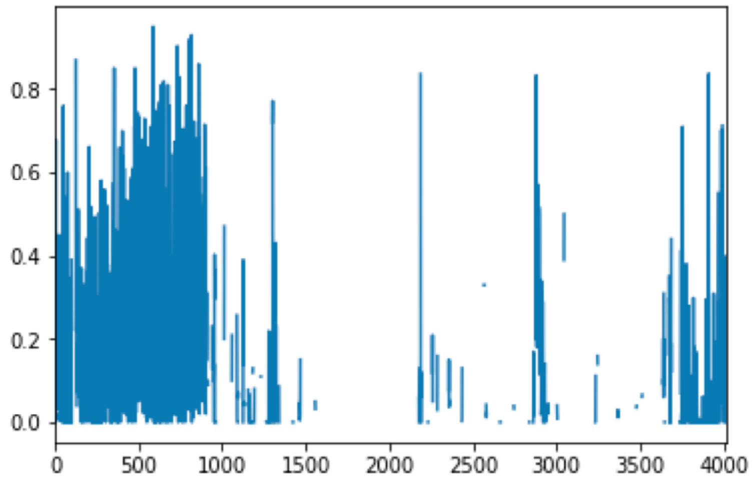
```
In [9]: # we can take min & maxes like with numpy arrays:
        # min and max of eccentricity
        planets['pl_orbeccen'].min(), planets['pl_orbeccen'].max()

Out[9]: (0.0, 0.95)
```

There are things like "groupby" and things that we aren't likely to get into right now, but will come across naturally later.

```
In [10]: # pandas also provides a matplotlib-like interface
         #  to make quick plots to look at our data
         planets["pl_orbeccen"].plot() # easy plots with pandas dataframes

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x108858b38>
```



Note this ils like doing a `matplotlib` style plot, but now our plot is associated with our data.

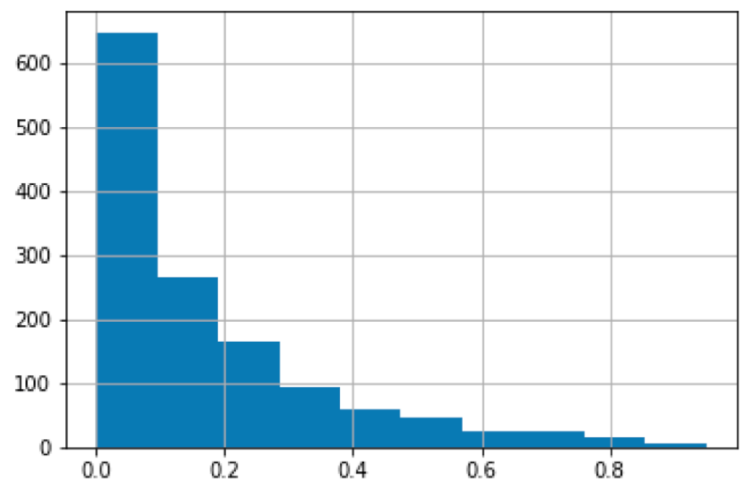
So we can see there are a lot of both zero and less very eccentric planets.

Note also that there are a lot of empty spots - this indicates where there are "NaN"s - or non-entries.

So, maybe this isn't what we want to know - we really want to know about the eccentricity *distribution* - so let's plot that:

```
In [14]: planets["pl_orbeccen"].hist()
```

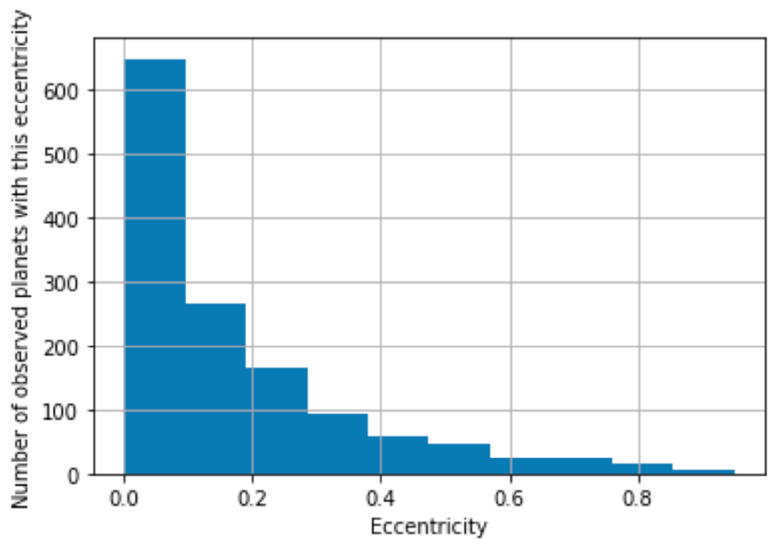
```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1213c6438>
```



We can add labels and things like this like we'd do with `matplotlib` type plots, but the way we do it is a little different:

```
In [17]: myPlot = planets["pl_orbeccen"].hist()
         myPlot.set_xlabel('Eccentricity')
         myPlot.set_ylabel('Number of observed planets with this eccentricity')
```

```
Out[17]: Text(0, 0.5, 'Number of observed planets with this eccentricity')
```



Now we are getting some where - it seems like there are many near-circular orbits in the Kepler dataset!

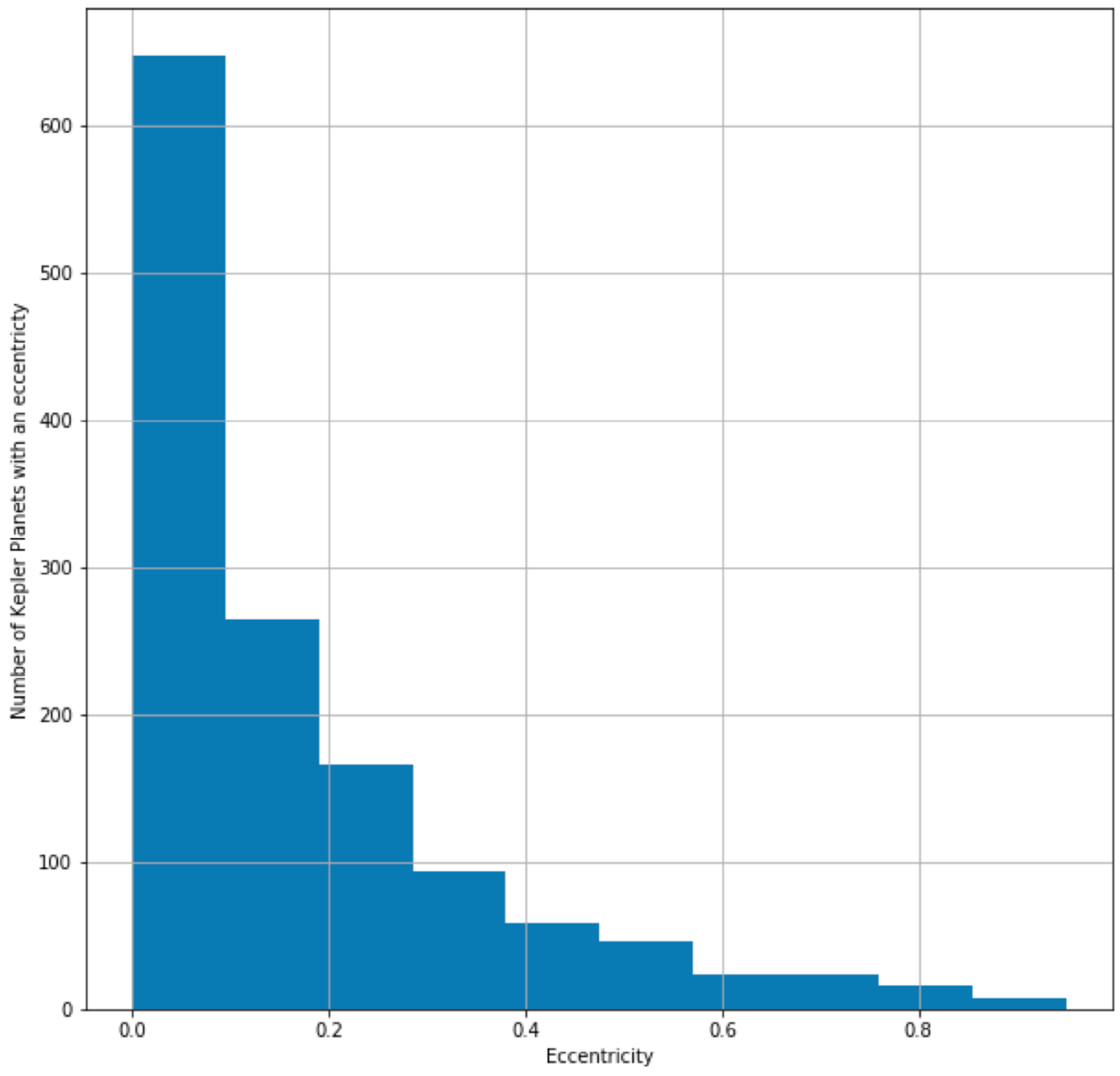
We can make the same sort of plot using `matplotlib` as well:

In [18]:
```
# first, create an axis object
fig, ax = plt.subplots(1, 1, figsize = (10, 10))

# set this histogram to be on this ax object
planets["pl_orbeccen"].hist(ax=ax)

# add labels with ax:
ax.set_xlabel('Eccentricity')
ax.set_ylabel('Number of Kepler Planets with an eccentricty')

plt.show()
```



Plot Styles

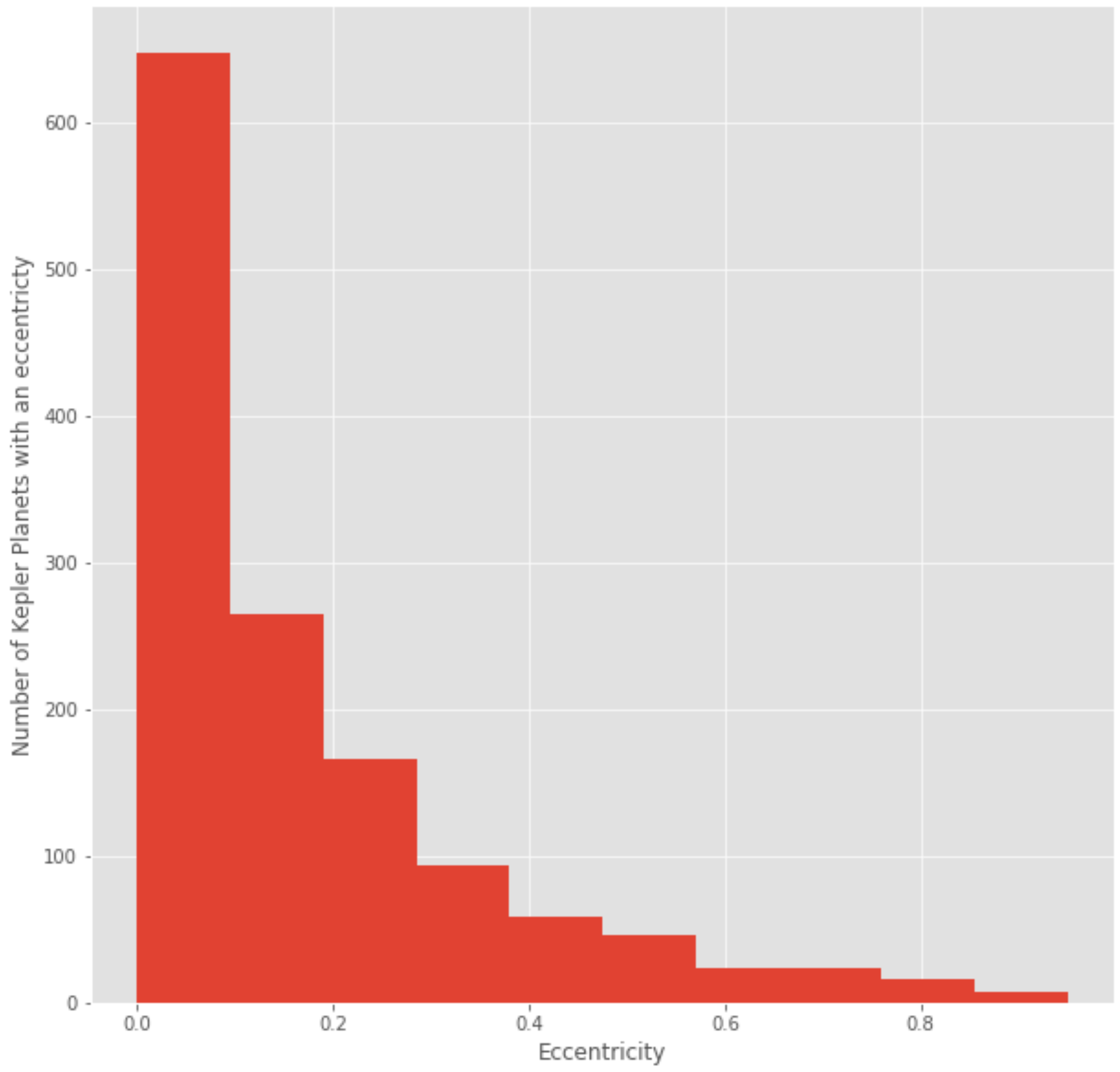
Now, lets play with something called the "style" of the plot. If you've used R, or come across it later in life you might/may have used the "ggplot" package. We can make plots in this style with Python too:

In [19]:
```
with plt.style.context("ggplot"):
    # first, create an axis object
    fig, ax = plt.subplots(1, 1, figsize = (10, 10))

    # set this histogram to be on this ax object
    planets["pl_orbeccen"].hist(ax=ax)

    # add labels with ax:
    ax.set_xlabel('Eccentricity')
    ax.set_ylabel('Number of Kepler Planets with an eccentricty')

    plt.show()
```



What plot styles are available to us?

In [20]:
```
plt.style.available
```

Out[20]:
```
['seaborn-dark',
 'seaborn-darkgrid',
 'seaborn-ticks',
 'fivethirtyeight',
 'seaborn-whitegrid',
 'classic',
 '_classic_test',
 'fast',
 'seaborn-talk',
 'seaborn-dark-palette',
 'seaborn-bright',
 'seaborn-pastel',
 'grayscale',
 'seaborn-notebook',
 'ggplot',
 'seaborn-colorblind',
 'seaborn-muted',
 'seaborn',
 'Solarize_Light2',
 'seaborn-paper',
 'bmh',
 'tableau-colorblind10',
 'seaborn-white',
 'dark_background',
 'seaborn-poster',
 'seaborn-deep']
```

But what if we want to see how our plot would look with each of these styles? We could just make a bunch of plots OR we can play with this interactively with ipywidgets:

In [21]:
```
import ipywidgets
```

In [23]:
```python
# lets tell jupyter ipywidgets that we want to
# mess around with the style of the plot
@ipywidgets.interact(style = plt.style.available)
def make_plot(style):
    with plt.style.context(style):
        # first, create an axis object
        fig, ax = plt.subplots(1, 1, figsize = (10, 10))

        # set this histogram to be on this ax object
        planets["pl_orbeccen"].hist(ax=ax)

        # add labels with ax:
        ax.set_xlabel('Eccentricity')
        ax.set_ylabel('Number of Kepler Planets with an eccentricty')

        plt.show()

# so now you can see that we get a little dropdown menu that lists
# all the different styles!
#  **play with this a bit!**
```

Ok, but what did we just do? We made something in Jupyter interactive. we'll have a lot of opportunities to mess around with widgets in the next class. If you want, you can read more on the docs: https://ipywidgets.readthedocs.io/en/stable/examples/Using%20Interact.html (https://ipywidgets.readthedocs.io/en/stable/examples/Using%20Interact.html)

Basically what ipywidgets.interact does is looks for a function with inputs and makes a little interactive option for those inputs, so like we did with the "make_plot" function above we can do for other things like change the value of a print statement:

```
@ipywidgets.interact(x=10)
def f(x):
    print("my value = " + str(x))
```

Note this is a little different to the format in the docs and you can use what you'd like the "@" symbol is a "decorator" and essentially its a way to sort of "extend" the interact function without modifying it to much.

At any rate, the take away is that you can call it like this, or how they do it in the docs, its up to you!

Another example: changing the histogram binning

We can actually run multiple widgets to do many interactive things at the same time, for example, plot style and histogram binning:

In [24]:
```
@ipywidgets.interact(style = plt.style.available, number_of_bins = range
(1,20,1))
def make_plot(style, number_of_bins):
    with plt.style.context(style):
        # first, create an axis object
        fig, ax = plt.subplots(1, 1, figsize = (10, 10))

        # set this histogram to be on this ax object
        planets["pl_orbeccen"].hist(ax=ax, bins=number_of_bins)

        # add labels with ax:
        ax.set_xlabel('Eccentricity')
        ax.set_ylabel('Number of Kepler Planets with an eccentricty')

        plt.show()
```

Exercise

Pick another variable besides eccentricity and repeat this exercise. Check out the header of the data file for lists of what the other parameters are.

Bonus: Pick another plotting variable besides style (like color of bars) to change. You might want to check up on the parameters avaiable under the pandas python plot: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.hist.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.hist.html) and general matplotlib plots: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.hist.html (https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.hist.html)

Bonus: make the variable another ipywidget input

Bonus: instead of a histogram, plot one variable vs. another

Bonus: make a 2 panel plot with the ability to change different things on different plots

In [ ]:
```

```