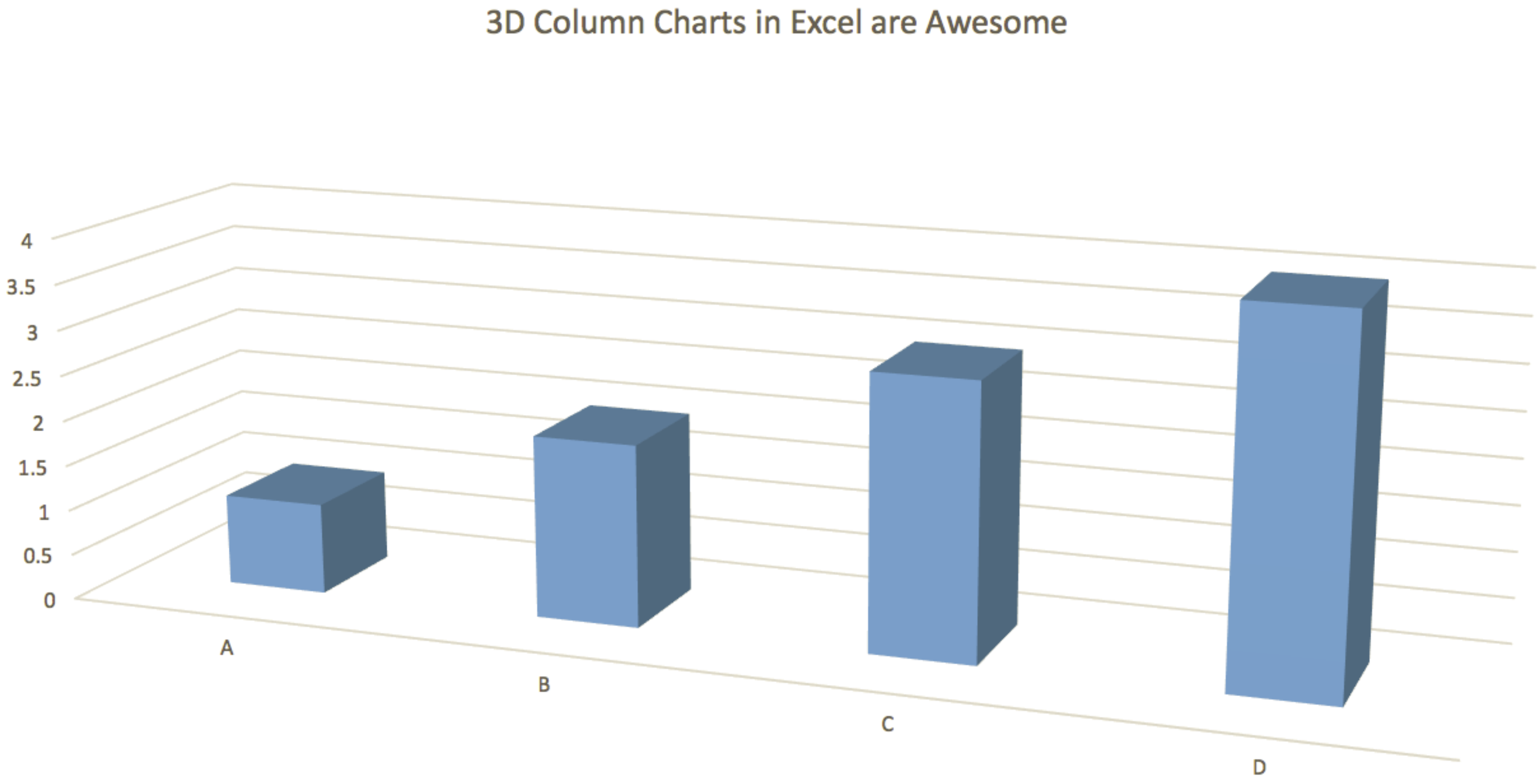
Modern Python plotting

Agenda

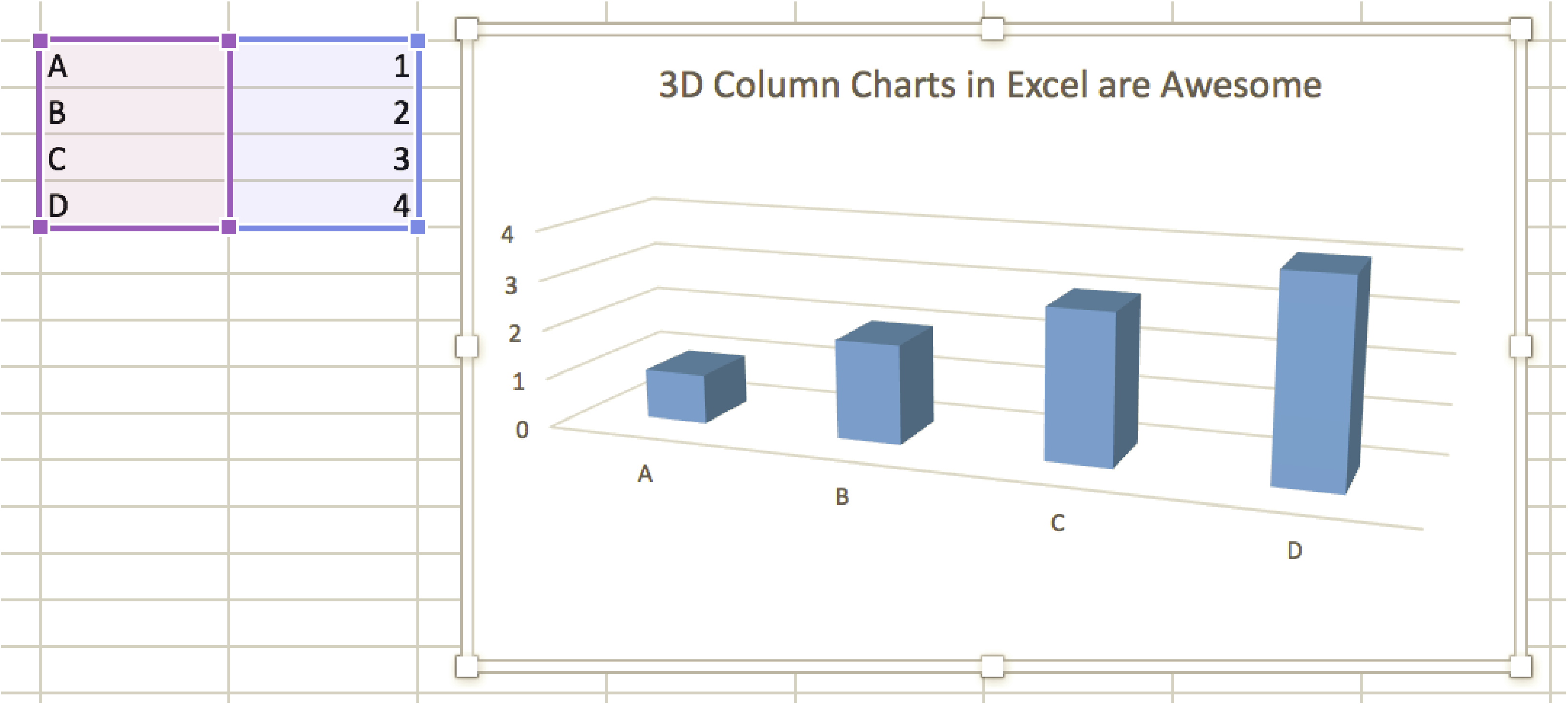
1. Basic exploratory plots with Pandas and Seaborn.
    - plots for single variables (histograms etc.)
    - plots for relationship between two or more variables (box, scatter, etc.)
2. Making explanatory plots useful and beautiful

Understanding plotting

What values do A,B,C,D have?



The shocking answer



A	1
B	2
C	3
D	4

What are you trying to accomplish?

1. Who's the audience?
    - Exploratory (use defaults) vs. explanatory (customize)
    - Raw data vs. model results
2. Graphs should be self explanatory
3. A graph is a narrative - should convey key point(s)

Analysis preparation

Getting prepared (1)

How do we start our analysis?

We first load our modules

```
In [ ]: import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import seaborn as sns

        %matplotlib inline
```

Getting prepared (2)

How do we load some data?

We load a standard dataset: tips.

```
In [ ]: tips = sns.load_dataset('tips')
```

Getting prepared (3)

How do we see what is in the DataFrame?

We get preview as follows:

```
In [ ]: print(tips)
```

Quiz: which variables/columns are available in the tips DataFrame?

Case: Plotting one numerical variable

From exploratory to final output

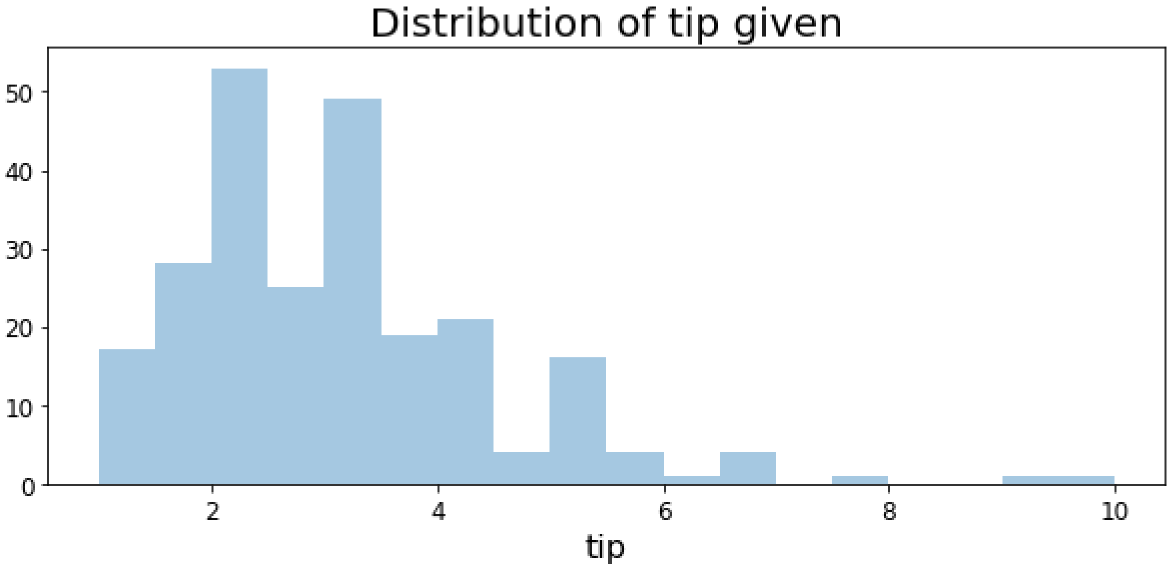
How do we plot the distribution of numerical variables?

We often use the histogram. Let's see what it is:

In [4]:

```
histplot
```

Out[4]:



Choosing your tool

In this course you will be exposed to several ways of plotting. All tools have their advantages.

Our options:

- the fundamental and flexible ~ matplotlib
- quick and dirty for long format ~ pandas
- a smart choice ~ seaborn

Histogram with matplotlib

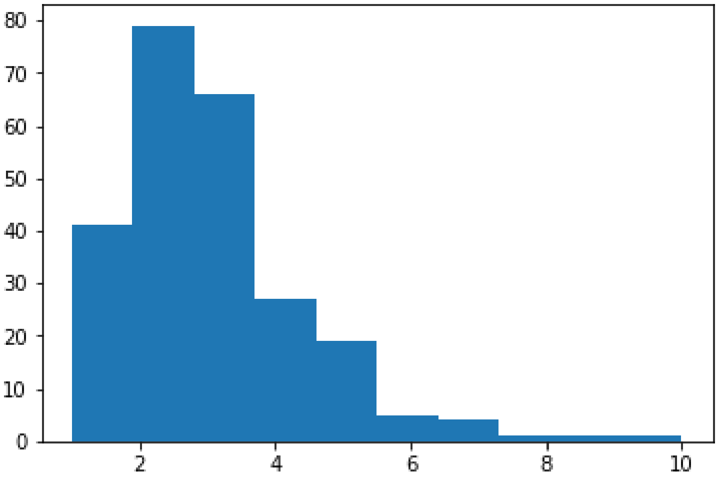
We will begin with the fundamental and flexible way. An old-school way of doing things.

In [4]:

```
f,ax = plt.subplots() # create placeholder for plot
ax.hist(tips.tip) # make plot
```

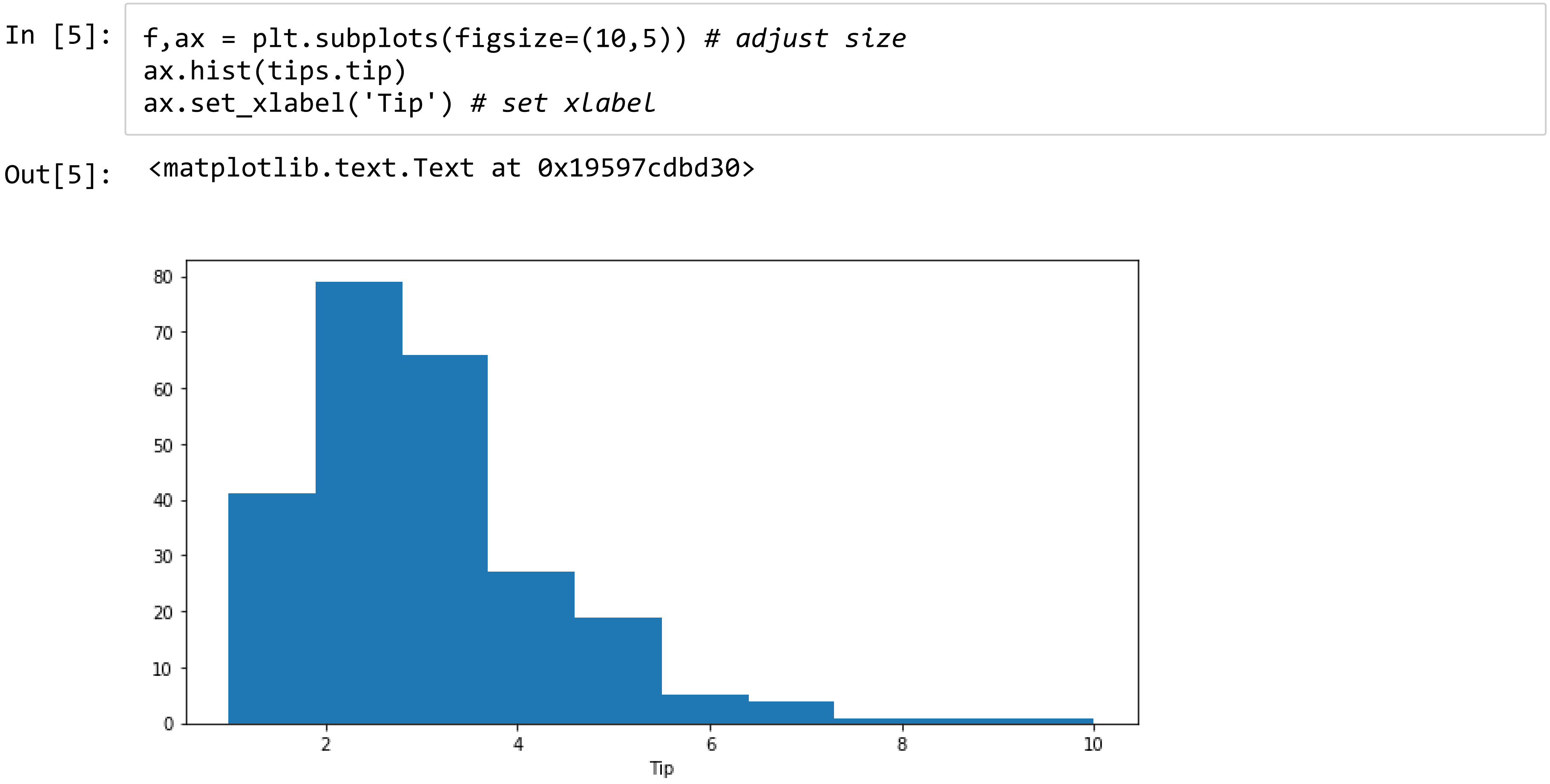
Out[4]:

```
(array([ 41.,  79.,  66.,  27.,  19.,   5.,   4.,   1.,   1.,   1.]),
 array([  1. ,   1.9,   2.8,   3.7,   4.6,   5.5,   6.4,   7.3,   8.2,
          9.1,  10. ]),
 <a list of 10 Patch objects>)
```



What might we change about this?

Examples of changes:



In [5]:
```
f,ax = plt.subplots(figsize=(10,5)) # adjust size
ax.hist(tips.tip)
ax.set_xlabel('Tip') # set xlabel
```

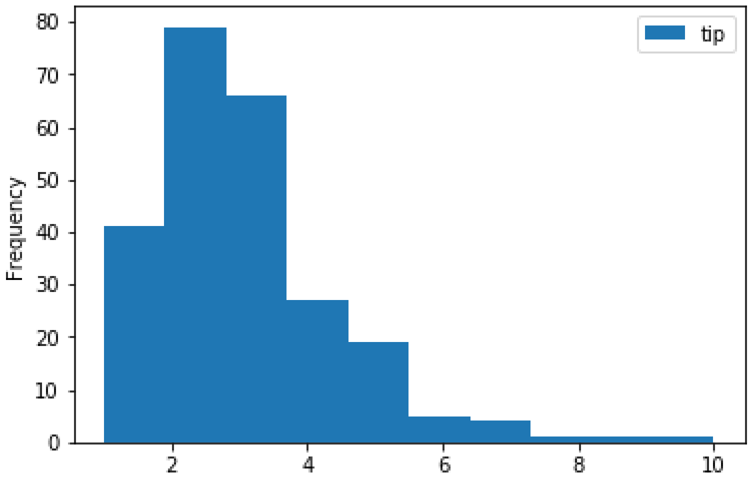
Out[5]:
```
<matplotlib.text.Text at 0x19597cdbd30>
```

Histogram - pandas

Pandas has a quick and dirty implemention. Let's try the code below.

```
In [9]: tips.plot(y='tip', kind='hist')
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x26f6304c080>
```

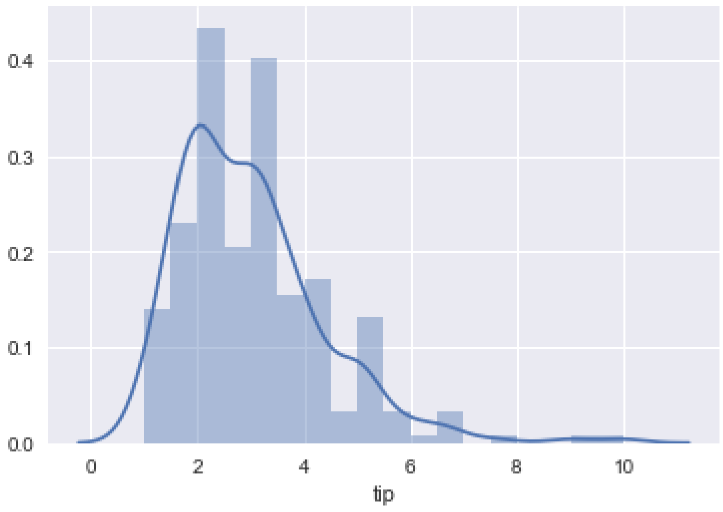


Histogram - seaborn

```
In [ ]: sns.set() # seaborn default
```

```
In [5]: sns.distplot(tips.tip) # make plot
```

```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x2569533b4e0>
```



What is the line?

Summing up

Group discussion (2 minutes):

- How did our tools perform?
- Which one seems most adequate for exploratory analysis?
- Which steps could be taken towards improving the figure?

Improving the histogram

What can be done change this histogram?

- How can we achieve the improvements?

Changing the figure size

In [ ]:

```
f,ax = plt.subplots(figsize=(10,6)) # set the plot size
sns.distplot(a=tips.tip,
             ax=ax) # use matplotlib defined plot for size)
```

Set title

In [ ]:

```
f,ax = plt.subplots(figsize=(10,6))
sns.distplot(a=tips.tip,
             ax=ax)
ax.set_title('Distribution of tips') # setting the title
```

Change bounds for x-axis

In [ ]:

```
f,ax = plt.subplots(figsize=(10,6))
sns.distplot(a=tips.tip,
             ax=ax)
ax.set_title('Distribution of tips')
ax.set_xlim(0,10) # set limits for x-axis
```

Add labels

In [ ]:

```
f,ax = plt.subplots(figsize=(10,6))
sns.distplot(a=tips.tip,
             ax=ax,
             kde_kws={'label': 'KDE'}, # label for KDE plot
             hist_kws={'label': 'Histogram'}) # label for histogram
ax.set_title('Distribution of tips')
ax.set_xlim(0,10)
```

Set font sizes

In [18]:

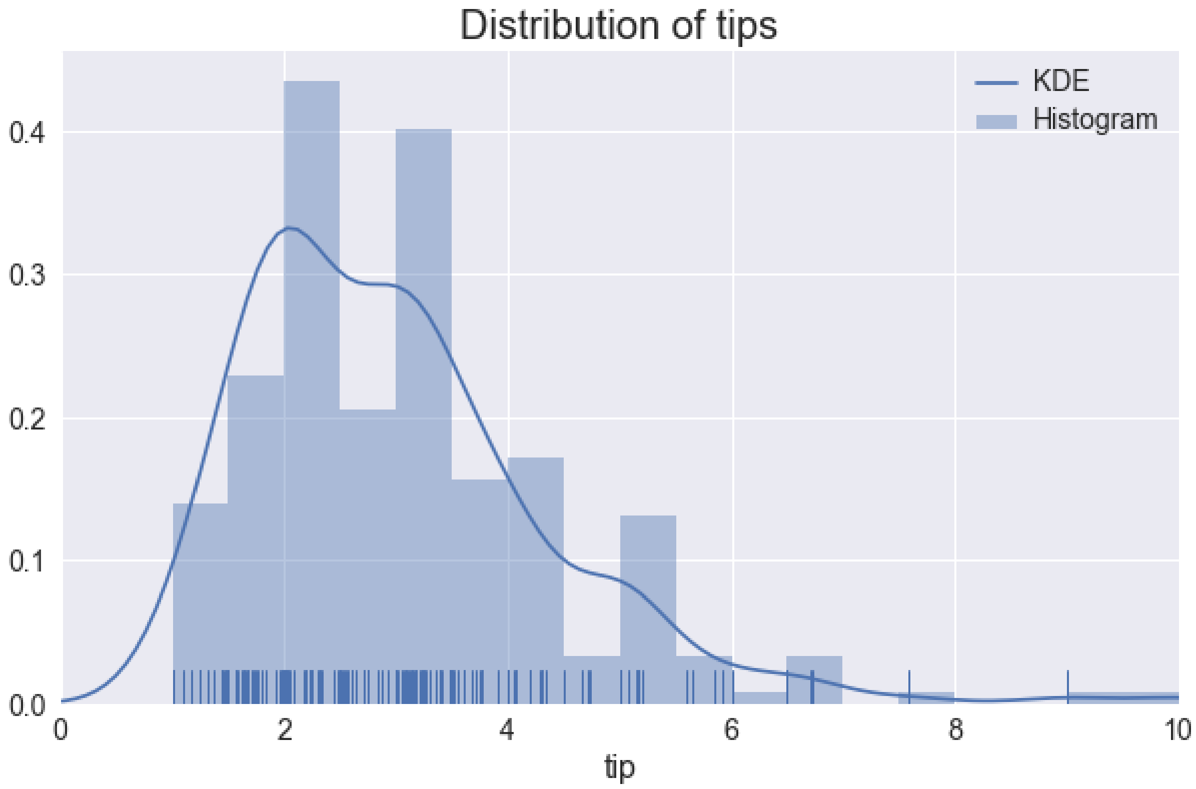
```
f,ax = plt.subplots(figsize=(10,6))
sns.distplot(a=tips.tip, ax=ax, rug=True,
             kde_kws={'label': 'KDE'},
             hist_kws={'label': 'Histogram'})
ax.set_title('Distribution of tips')
ax.set_xlim(0,10)

# set font sizes
ax.title.set_fontsize(20) # title
ax.xaxis.label.set_fontsize(16)  #xaxis label
for item in ax.get_yticklabels()+ax.get_xticklabels(): # xaxis tickers
    item.set_fontsize(14)
plt.setp(plt.gca().get_legend().get_texts(), fontsize='14') # legend labels
print()
```

The final plot

In [5]:

```
f
```

Out[5]:



Explanation for the final plot

In [ ]:

```
f,ax = plt.subplots(figsize=(10,6)) # set the plot size
sns.distplot(a=tips.tip,
             ax=ax, # use matplotlib defined plot for size
             rug=True, # include raw count
             kde_kws={'label': 'KDE'}, # label for KDE plot
             hist_kws={'label': 'Histogram'}) # label for histogram
ax.set_title('Distribution of tips') # set title
ax.set_xlim(0,10) # set x limits

# set font sizes
ax.title.set_fontsize(20) # title
ax.xaxis.label.set_fontsize(16)  #xaxis label
for item in ax.get_yticklabels()+ax.get_xticklabels(): # xaxis tickers
    item.set_fontsize(14)
plt.setp(plt.gca().get_legend().get_texts(), fontsize='14') # legend labels
print()
```

Setting - standard plot size

```
In [43]: plt.rcParams['figure.figsize'] = 10,6
```

Univariate categorical data

What if we have categorical data?

What is categorical data? Example gender count:

In [ ]:

```
count_sex = tips.sex.value_counts()
count_sex
```

Let's plot this with bars:

```
In [ ]: count_sex.plot.bar()
```

(click down for pie chart)

Let's plot this as a pie:

In [ ]:

```
count_sex.plot.pie()
```

Generate more data

How do we generate timeseries data?

We create some data

```
In [5]: np.random.seed(123) # set seed - then we make some random data

        ts = np.random.normal(0,1,[1000,3]) # time series with no slope

        dates = pd.date_range(start='20170801', periods=1000, freq='D') # dates for variables
```

We create a dataset with time series.

```
In [6]: df = pd.DataFrame(data=ts,
                          index=dates,
                          columns=['A', 'B', "C"])\
               .cumsum()

        df['A'] += np.arange(0,60,.06)
        df['B'] += np.arange(0,30,.03)
```

Power of Pandas

Why is pandas used in fin-tech so much?

Example: Plotting time series for one variable

In [ ]:

```
df.A.plot()
```

Canonical table formats

How do we define a tidy/long table?

One row for each observation

Quiz: *Is our DataFrame, df, wide?*

Plotting multiple variables

Wide formatting

Which tool should we pick for wide data?

Pandas!

Plotting time series

How do we plot multiple time series in one plot?

In [ ]:

```
df.plot()
```

The boxplot

```
In [ ]: df.plot.box()
```

Histogram with multiple variables:

```
In [ ]: df.plot.hist(alpha=.5)
```

The scatter plot

In [ ]:

```
df.plot.scatter(x='A',y='B')
```

Quiz: How might we alter the scatter plot?

- Let's try to change the colors of the dots:

```
In [ ]: df.plot.scatter(x='A',y='B',c='C')
```

Seaborn for scatter and related

- The jointplot for scatter

In [ ]:

```
sns.jointplot('A','B',data=df, joint_kws={'alpha':0.3})
```

How can we modify this? KDE, hexbin?

- The regression plot

```
In [ ]: sns.lmplot('A','B',data=df)
```

- Multiple scatterplots (correlation matrix style)

```
In [ ]: sns.pairplot(df)
```

Plotting multiple variables

Using long format

What was long format? (one row per observation)

What columns can we use as extra info? Categorical variables?

Let's make a boxplot of tips - distinguish by smoker:

```
In [ ]:
```

Let's try a barplot of tips. Distinguish in addition by gender:

In [ ]:

Data exploration

The FacetGrid

In [ ]:

```
g = sns.FacetGrid(tips)
g = g.map(sns.regplot, 'total_bill','tip')
```

Let's try to add gender distinctive slopes

```
In [ ]: g = sns.FacetGrid(tips, row = 'sex')
        g = g.map(sns.regplot, 'total_bill','tip')
```

Let's try to further add seperate estimates for smoking status

```
In [ ]:
```

Can we say anything about smokers tipping behavior?

Summing up

Exploratory analysis

- We learned how we could leverage Pandas for data in wide format.
- We learned that Seaborn can make great initial visualization and is a powerful tool for exploration of data.

Explanatory plots

- This involves extra work to have camera ready.
- Matplotlib must be configured.

If you want learn more

Other useful plots can be found in the tutorials of Seaborn (https://seaborn.pydata.org/).

To master plot making in python the tweaking with matplotlib (https://matplotlib.org/) is essential. Advice: google is your friend when searching for how to configure a plot.