Data Visualization with Python

In Jupyter Notebook!

Today we will

- Play with Jupyter Notebooks
- Import data into a pandas data frame
- Import some standard and useful libraries for python
- Visualize and describe data with box plots, histograms, scatter plots, and descriptive statistics

Reference and Resource

This lesson and data is adapted from [LinkedIn Learning: Python Statistics Essential Training (https://www.linkedin.com/learning/python-statistics-essential-training/)](https://www.linkedin.com/learning/python-statistics-essential-training/). See these lessons for more details including working with categorical data.

Playing with Jupyter Notebooks

CELLS - Markdown versus Code

This is a markdown cell. It renders text as HTML.

I can type in **bold**

- I can have bullet points

I can add LaTex $\sqrt{2+3^8}$

In [1]:
```
# This is a code cell
# We will add and run python in code cells
message = 'Hello World'
print(message)
```

```
Hello World
```

Importing Libraries and Data

In [2]:
```
# Load standard libraries for data analysis
# When we use "as", we are naming an alias for the library name
import numpy as np
import pandas as pd

import matplotlib
import matplotlib.pyplot as pp

import scipy.stats

# To render plots inline, we use this Jupyter Notebook "magic" command
%matplotlib inline
```

In [3]:
```
# Have a question about a package?
# Get documentation with the question mark ?
# INSTRUCTIONS:  Ask about a library here:

?scipy.stats
```

Data Cleanup

Come with me for a quick sideline to planets.xls!

Welcome back . . . let's read in the dataframe.

In [4]:
```
#  Use pandas to read in our comma-separated value dataframe (i.e. tabl
e)
#      where the cases are in rows and the variables (or attributes) in c
olumns
#      There is quantatitive and categorical data!

# INSTRUCTIONS:  Add the filename.  You can use tab to complete a filena
me.
planets = pd.read_csv('Planets.csv')
```

In [5]:
```
# INSTRUCTIONS:  Uncomment array name to display the data that was read
 in
# planets
```

In [6]:
```
# What if we only want the first couple columns of data?
planets = pd.read_csv('Planets.csv', usecols=[0,1,2,3,])
planets
```

Out[6]:

	Planet	Mass	Diameter	DayLength
0	MERCURY	0.3300	4879	4222.6
1	VENUS	4.8700	12,104	2802.0
2	EARTH	5.9700	12,756	24.0
3	MOON	0.0730	3475	708.7
4	MARS	0.6420	6792	24.7
5	JUPITER	1898.0000	142,984	9.9
6	SATURN	568.0000	120,536	10.7
7	URANUS	86.8000	51,118	17.2
8	NEPTUNE	102.0000	49,528	16.1
9	PLUTO	0.0146	2370	153.3

In [7]:
```
planets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Planet     10 non-null     object
 1   Mass       10 non-null     float64
 2   Diameter   10 non-null     object
 3   DayLength  10 non-null     float64
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

In [8]:
```
# View data in row 0
planets.iloc[0]
```

Out[8]:
```
Planet        MERCURY
Mass             0.33
Diameter         4879
DayLength      4222.6
Name: 0, dtype: object
```

In [32]:
```
planets.columns
```

```
Object `plot` not found.
```

In [10]:
```
#  INSTRUCTIONS:  Look at planet masses,
# a basic pandas object called a Series
# planets['Mass']
# We can also type it planets.Mass
```

In [11]:
```
# Notice the range of the rows
planets.index
```

Out[11]: RangeIndex(start=0, stop=10, step=1)

In [12]:
```
# Fix indexing so we can see planet names instead of numeric range
# Use the method set_index on the dataframe object
planets.set_index('Planet')
```

Out[12]:

Planet	Mass	Diameter	DayLength
MERCURY	0.3300	4879	4222.6
VENUS	4.8700	12,104	2802.0
EARTH	5.9700	12,756	24.0
MOON	0.0730	3475	708.7
MARS	0.6420	6792	24.7
JUPITER	1898.0000	142,984	9.9
SATURN	568.0000	120,536	10.7
URANUS	86.8000	51,118	17.2
NEPTUNE	102.0000	49,528	16.1
PLUTO	0.0146	2370	153.3

In [13]:
```
# This results in a copy of the dataframe object.
planets
```

Out[13]:

	Planet	Mass	Diameter	DayLength
0	MERCURY	0.3300	4879	4222.6
1	VENUS	4.8700	12,104	2802.0
2	EARTH	5.9700	12,756	24.0
3	MOON	0.0730	3475	708.7
4	MARS	0.6420	6792	24.7
5	JUPITER	1898.0000	142,984	9.9
6	SATURN	568.0000	120,536	10.7
7	URANUS	86.8000	51,118	17.2
8	NEPTUNE	102.0000	49,528	16.1
9	PLUTO	0.0146	2370	153.3

In [14]:
```
# To modify the original we use inplace
planets.set_index('Planet',inplace=True)
```

In [15]:
```
# See the original has updated range names now.
planets
```

Out[15]:

	Mass	Diameter	DayLength
Planet			
MERCURY	0.3300	4879	4222.6
VENUS	4.8700	12,104	2802.0
EARTH	5.9700	12,756	24.0
MOON	0.0730	3475	708.7
MARS	0.6420	6792	24.7
JUPITER	1898.0000	142,984	9.9
SATURN	568.0000	120,536	10.7
URANUS	86.8000	51,118	17.2
NEPTUNE	102.0000	49,528	16.1
PLUTO	0.0146	2370	153.3

In [16]:
```
planets.iloc[0]
```

Out[16]:
```
Mass              0.33
Diameter          4879
DayLength       4222.6
Name: MERCURY, dtype: object
```

In [17]:
```
planets.loc['MERCURY']
```

Out[17]:
```
Mass              0.33
Diameter          4879
DayLength       4222.6
Name: MERCURY, dtype: object
```

In [18]:
```
# There are lots of smart indexing ways to access data.  For example
# INSTRUCTIONS: Uncomment and ctrl+Enter to test!
#planets.Mass['EARTH']
#planets.loc['EARTH'].Mass
#planets.loc['EARTH','Mass']
```

Descriptive statistics

In [19]:
```
# Let's check some simple descriptive statistics
# min(), max(), mean(), var(), quantiles()
planets.Mass.min()
```

Out[19]:
```
0.0146
```

In [20]:
```
planets.Mass.max()
```
Out[20]:
```
1898.0
```

In [21]:
```
planets.Mass.mean()
```
Out[21]:
```
266.66996000000006
```

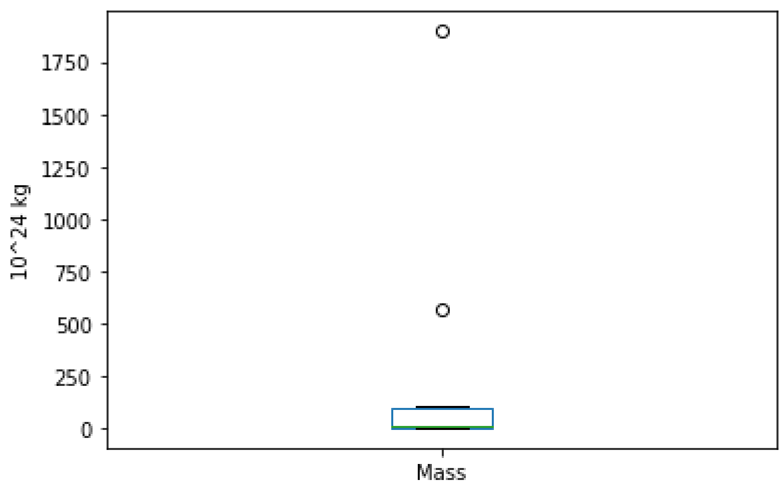
In [22]:
```
planets.Mass.var()
```
Out[22]:
```
359099.71652690484
```

In [23]:
```
planets.mean()
```
Out[23]:
```
Mass           266.66996
DayLength      798.92000
dtype: float64
```

In [24]:
```
planets.Mass.quantile([0.25,0.50,0.75])
```
Out[24]:
```
0.25     0.408
0.50     5.420
0.75    98.200
Name: Mass, dtype: float64
```

In [25]:
```
planets.mean()
```
Out[25]:
```
Mass           266.66996
DayLength      798.92000
dtype: float64
```

In [26]:
```
planets.Mass.plot(kind='box')
pp.ylabel('10^24 kg')
```
Out[26]:
```
Text(0, 0.5, '10^24 kg')
```



DESCRIBE AND PLOT DISTRIBUTIONS

For this, we will import some more librariers and a richer dataset from GapMinder (https://www.gapminder.org)

In [27]:
```
# Import Librariers
import numpy as np
import scipy.stats
import pandas as pd

import matplotlib
import matplotlib.pyplot as pp

from IPython import display
from ipywidgets import interact, widgets

%matplotlib inline

import re
import mailbox
import csv
```

In [28]:
```
# Import data
gapminder = pd.read_csv('gapminder.csv')
```

In [29]:
```
gapminder.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14740 entries, 0 to 14739
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   country            14740 non-null  object 
 1   year               14740 non-null  int64  
 2   region             14740 non-null  object 
 3   population         14740 non-null  float64
 4   life_expectancy    14740 non-null  float64
 5   age5_surviving     14740 non-null  float64
 6   babies_per_woman   14740 non-null  float64
 7   gdp_per_capita     14740 non-null  float64
 8   gdp_per_day        14740 non-null  float64
dtypes: float64(6), int64(1), object(2)
memory usage: 1.0+ MB
```

In [30]:
```
gapminder
```

Out[30]:

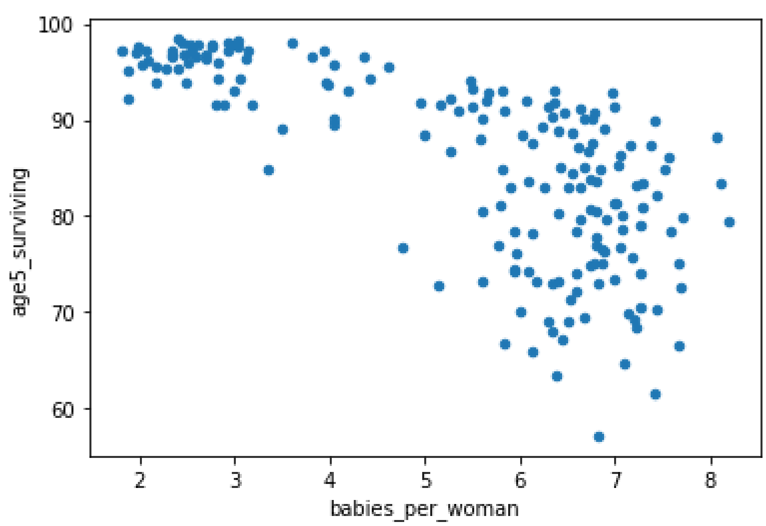
	country	year	region	population	life_expectancy	age5_surviving	babies_per_woman
0	Afghanistan	1800	Asia	3280000.0	28.21	53.142	7.00
1	Afghanistan	1810	Asia	3280000.0	28.11	53.002	7.00
2	Afghanistan	1820	Asia	3323519.0	28.01	52.862	7.00
3	Afghanistan	1830	Asia	3448982.0	27.90	52.719	7.00
4	Afghanistan	1840	Asia	3625022.0	27.80	52.576	7.00
...	...	...	...	...	...	...	...
14735	Zimbabwe	2011	Africa	14255592.0	51.60	90.800	3.64
14736	Zimbabwe	2012	Africa	14565482.0	54.20	91.330	3.56
14737	Zimbabwe	2013	Africa	14898092.0	55.70	91.670	3.49
14738	Zimbabwe	2014	Africa	15245855.0	57.00	91.900	3.41
14739	Zimbabwe	2015	Africa	15602751.0	59.30	92.040	3.35

14740 rows × 9 columns

Scatter plots

In [31]:
```
# Lets look at a scatter plot of a subset of the data
gapminder[gapminder.year == 1965].plot.scatter('babies_per_woman','age5_
surviving')
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x11b001110>



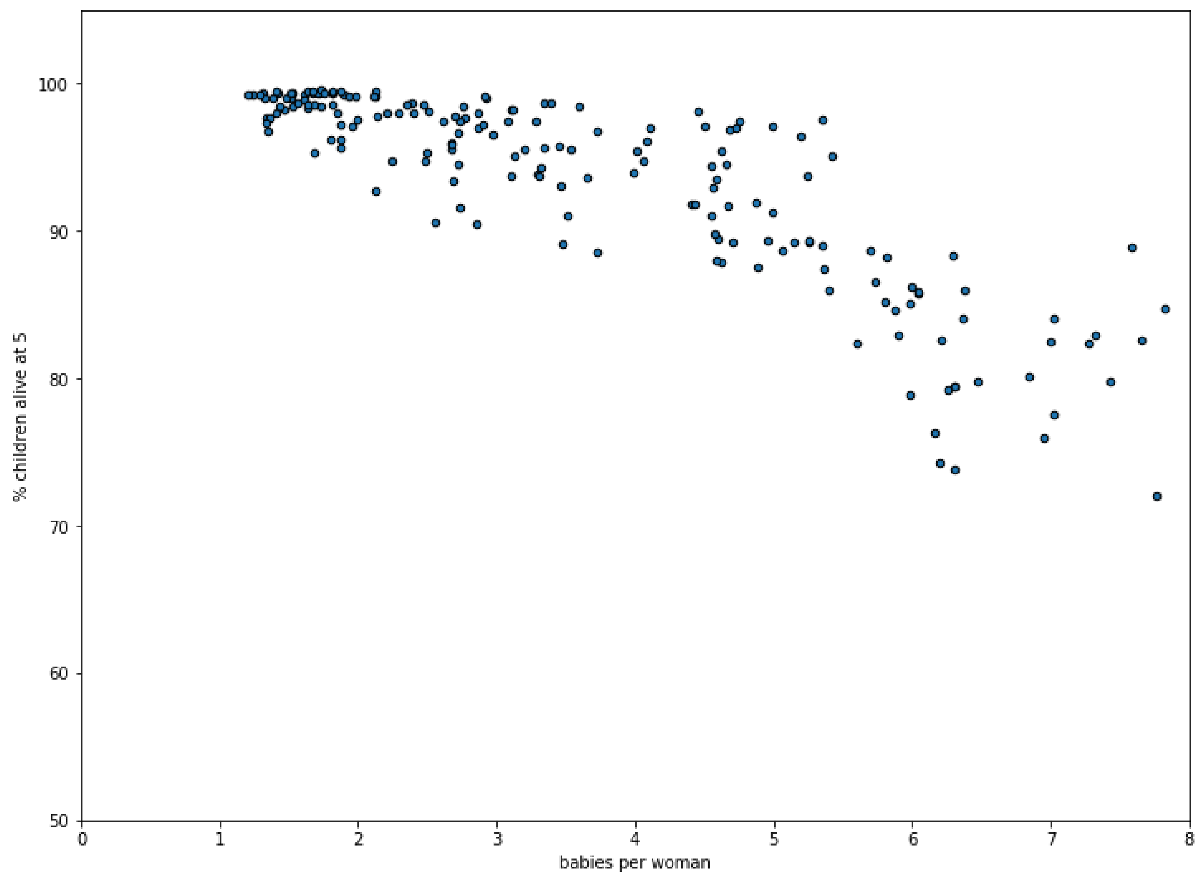
In [33]:
```
#  Define a function to plot a scatter plot of a year
#  of data for babies_per_women to age5_surviving

def plotyear(year):
    data = gapminder[gapminder.year == year]
#    area = 5e-6 * data.population
#    colors = data.region.map({'Africa': 'skyblue', 'Europe': 'gold', 'A
merica': 'palegreen', 'Asia': 'coral'})

    data.plot.scatter('babies_per_woman','age5_surviving',
#                      s=area,c=colors,
                      linewidths=1,edgecolors='k',
                       figsize=(12,9))

    pp.axis(ymin=50,ymax=105,xmin=0,xmax=8)
    pp.xlabel('babies per woman')
    pp.ylabel('% children alive at 5')
```

In [34]:
```
plotyear(1995)
```



In [35]:
```
# Add a very cool widget
interact(plotyear,year=widgets.IntSlider(min=1950,max=2015,step=1,value=
1965))
```

Out[35]:
```
<function __main__.plotyear(year)>
```

Histogram

Let's look at the distribution of global life expectancies in a certain year.

In [36]:
```
gapminder.head()
```

Out[36]:

	country	year	region	population	life_expectancy	age5_surviving	babies_per_woman	gdp
0	Afghanistan	1800	Asia	3280000.0	28.21	53.142	7.0	
1	Afghanistan	1810	Asia	3280000.0	28.11	53.002	7.0	
2	Afghanistan	1820	Asia	3323519.0	28.01	52.862	7.0	
3	Afghanistan	1830	Asia	3448982.0	27.90	52.719	7.0	
4	Afghanistan	1840	Asia	3625022.0	27.80	52.576	7.0	

In [37]:
```
data = gapminder[gapminder.year==2015]
```

In [38]:
```
data.head()
```

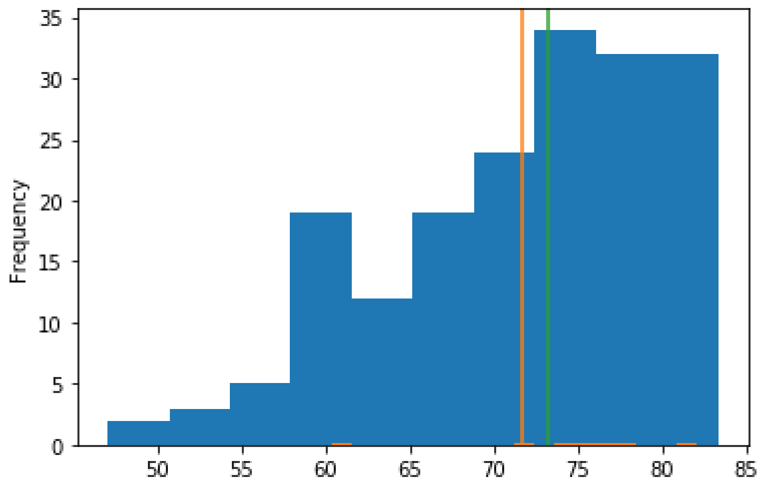
Out[38]:

	country	year	region	population	life_expectancy	age5_surviving	babies_per_woman
80	Afghanistan	2015	Asia	32526562.0	53.8	90.89	4.47
161	Albania	2015	Europe	2896679.0	78.0	98.60	1.78
242	Algeria	2015	Africa	39666519.0	76.4	97.60	2.71
323	Angola	2015	Africa	25021974.0	59.6	84.31	5.65
404	Antigua and Barbuda	2015	America	91818.0	76.4	99.19	2.06

In [44]:
```
data.life_expectancy.plot(kind='hist')
# We can assign number of bins, and normalize
data.life_expectancy.plot(kind="hist",bins=30,density=True)

# We can add lines at the mean and median
#pp.axvline(data.life_expectancy.mean(),c='C1')
#pp.axvline(data.life_expectancy.median(),c='C2')
```

Out[44]: <matplotlib.lines.Line2D at 0x1a1fb2dc50>



In [45]:
```
# We can also find the percentile of countries with < certain life expec
tancy
scipy.stats.percentileofscore(data.life_expectancy,75)
```

Out[45]: 59.34065934065934

More Scatter plots

In [46]:
```
data = gapminder[gapminder.country=='United States']
data
```

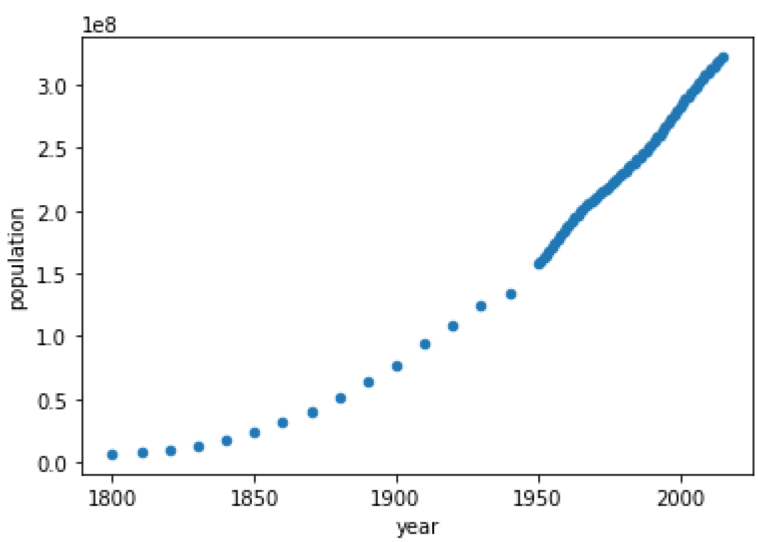
Out[46]:

	country	year	region	population	life_expectancy	age5_surviving	babies_per_woman
14011	United States	1800	America	6801854.0	39.41	53.711	7.03
14012	United States	1810	America	8294928.0	39.41	53.904	6.81
14013	United States	1820	America	10361646.0	39.41	54.443	6.59
14014	United States	1830	America	13480460.0	39.41	55.406	6.38
14015	United States	1840	America	17942443.0	39.41	57.383	6.18
...	...	...	...	...	...	...	...
14087	United States	2011	America	312390368.0	78.90	99.280	1.90
14088	United States	2012	America	314799465.0	79.00	99.290	1.90
14089	United States	2013	America	317135919.0	79.10	99.310	1.98
14090	United States	2014	America	319448634.0	79.10	99.330	1.97
14091	United States	2015	America	321773631.0	79.10	99.350	1.97

81 rows × 9 columns

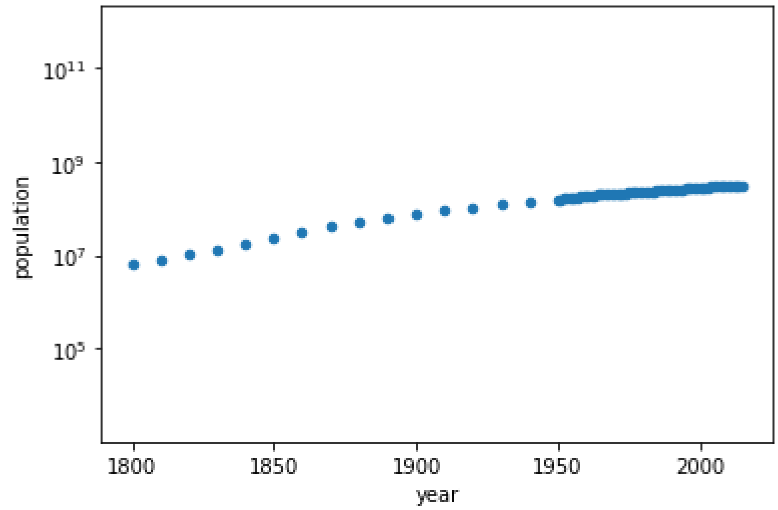
In [47]:
```
data.plot.scatter('year','population')
```

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1fb068d0>



In [48]:
```
data.plot.scatter('year','population', logy=True)
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x1098d60d0>



In [49]:
```
# Let's get data for two countries to compare.
data = gapminder.query('(country == "Italy") or (country == "United Stat
es")')
```

In [50]:
```
color = np.where(data.country=='Italy','blue','orange')
data.plot.scatter("year","population",c=color)
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1e4a3290>