import matplotlib.pyplot as plt

%matplotlib inline

import numpy as np

xs = np.linspace(0, 10, 100)

ys = np.sin(xs)

plt.plot(xs, ys[,'r.-'])=> r=color red, .-=line type

plt.scatter(xs,ys) => axes=df.scatter('col as x', 'col as y'[,alpha=0.1])

alpha value used for blending (0=transparent, 1=opaque)

plt.bar(xs,ys)

plt.hexbin(…)=hexagonal binning plot.

plt.boxplot( )=compare disstributions sside by side

plt.hist(data,bins=N, histtype=T) = default bins-10

histtype : {'bar', 'barstacked', 'step', 'stepfilled'},

f=plt.hist(df['column'],range=(-100,300),bins=500,histtype='step')

this is the same: axes = df['column'].hist(bins=500)

axes.set\_xlim(-100, 300)

plt.title('smth')=> over the plot just made

axes.set\_xlim(min, max) .set\_ylim(min,max)

plt.subplot(221)= num of plots=2x2, activating subplot 1 (count from 1)

plt.show()= has to follow sublot to show it

fig, subplotarray = plt.subplots(2,3) = figure with subplots prepared

subplotarray[x][y].scatter(xs, ys) -> count as in array from 0

plt.ylabel(r'$\sum log(n^2)$ (not really)') => using latex

plt.xlabel('normal name')

plt.ylim(min,max) ….. plt.xlim(min, max)

plt.yscale('log') plt.xscale('log') ->linear'/'log'/'logit'/'symlog'

plt.title('TITLE')

delays, hours = [], []

for hour, hourly\_df in by\_hour:

delays.append(hourly\_df['DepDelay'])

hours.append(hour)

labels = map(lambda f: '%02d' % f, hours)

plt.boxplot(delays, labels=labels, positions=hours)

plt.ylim(-50, 200)

import seaborn as sns

change matplotlib defaults and make graphs prettier

axes = sns.boxplot(x='Hour', y='DepDelay', data=df)

sns.kdeplot(df['col'])= Fit&plot univariate or bivariate density estimate.

\_ =sns.distplot(df[' selling price ']) = combines the matplotlib ``hist`` (automatic bin size calc) with the seaborn `kdeplot` and `rugplot`

sns.kdeplot(data, data2=None, shade=False, vertical=False, kernel='gau', bw='scott', gridsize=100, cut=3, clip=None, legend=True, cumulative=False, shade\_lowest=True, ax=None, \*\*kwargs)

kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable. The bandwith in this context is the width of the individual distribution functions we fit. The greater, the smoother is the density estimate.

sns.jointplot(x = 'col1', y = 'col2', data = df, kind='kde')

joinplot allows to examine the covariance of two variables

kind : { "scatter" | "reg" | "resid" | "kde" | "hex" },

Web API (aplication programming interface) import requests

resp = requests.get('url')

resp.content= string content

resp.status\_code

madrid\_coords = {'lat': 40.4, 'lon': -3.7}

r = requests.get('http://api.open-notify.org/iss-pass.json', params=madrid\_coords)

JSON import json

pyt\_obj = json.loads(json\_string)

json\_str=json.dumps(pyt\_object)

from IPython.display import IFrame, HTML

**IFrame**('readme.html', 800, 600)

IFrame('http://www.elpais.es', 800, 600)

from zipfile import ZipFile

zip\_file = **ZipFile**(path\_to\_one\_file)= open a connection with zip file

files\_inside=**zip\_file.filelist** = get connectors to all files in zip

file1, file2=zip\_file.filelist (if we know there are 2 files)

files\_inside[N]**.filename** = show where the link is connected

f=**zip\_file.open**(**zip\_file.filelist[0].filename**)

open file like with normal open 🡪 f.next() f.read() df=pd.read\_csv(f)

Web Scraping from bs4 import BeautifulSoup

Page=requests.get('url').content

soup = BeautifulSoup(page, 'lxml')

print soup.prettify()[:1000]

alerts = soup.find\_all('div', class\_='ec\_statements')

Extracts a list of Tag objects that match the given

criteria. You can specify the name of the Tag and any

attributes you want the Tag to have.

home/dsc/anaconda2/bin/jupyter notebook

conda install lxml

conda install BeautifulSoup