master matplotlib / MPL Tut.ipynb Go to file ...

derekbanas Add files via upload ... Latest commit dfaf765 on Aug 22, 2020 History

1 contributor

1076 lines (1076 sloc) | 720 KB

Raw Blame

# Matplotlib Tutorial

Matplotlib provides numerous ways to create static, animated and interactive visualizations. It is the most popular plotting library for Python. It works easily with both NumPy and Pandas arrays. Seaborn, which I'll cover in my next tutorial futher extends Matplotlib, but it is very important to learn both.

You can install it with the command : conda install matplotlib or pip install matplotlib

https://matplotlib.org/gallery/index.html (https://matplotlib.org/gallery/index.html) is a great page to go to when you are looking for information on making a specific chart type.

## Import

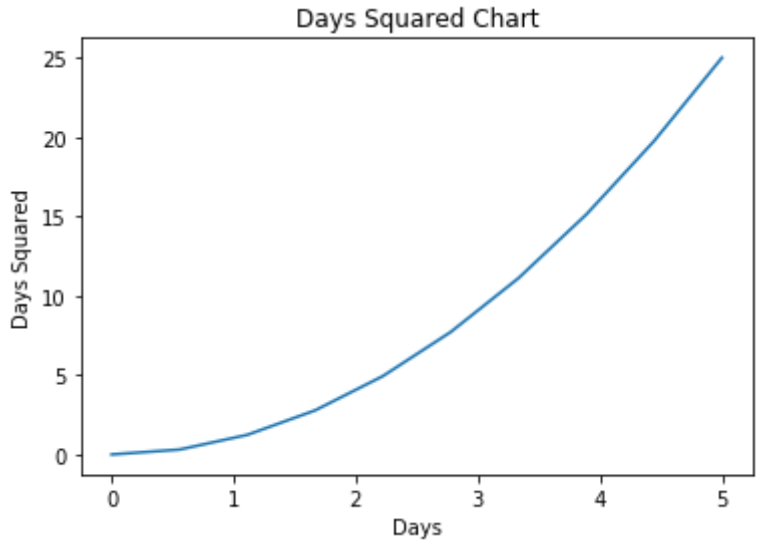
In [403]:
```
# Import Matplotlib and allow plots to show in the Jupyter Notebook
import matplotlib.pyplot as plt
%matplotlib inline
# Import NumPy and Pandas
import numpy as np
import pandas as pd

# Auto reloads notebook when changes are made
%reload_ext autoreload
%autoreload 2
```

## Functional Plot

In [404]:
```
# Create x & y NumPy arrays (10 Floats from 0 to 5)
# Any NumPy array will do for x & y points as long as you have
# an equal number
x_1 = np.linspace(0,5,10)
y_1 = x_1**2
# Display plot with x & y
plt.plot(x_1, y_1)
# Add a title
```

```
plt.title('Days Squared Chart')
# Add an X & Y Label
plt.xlabel('Days')
plt.ylabel('Days Squared')
# If not in Jupyter Notebook use
# plt.show()
```

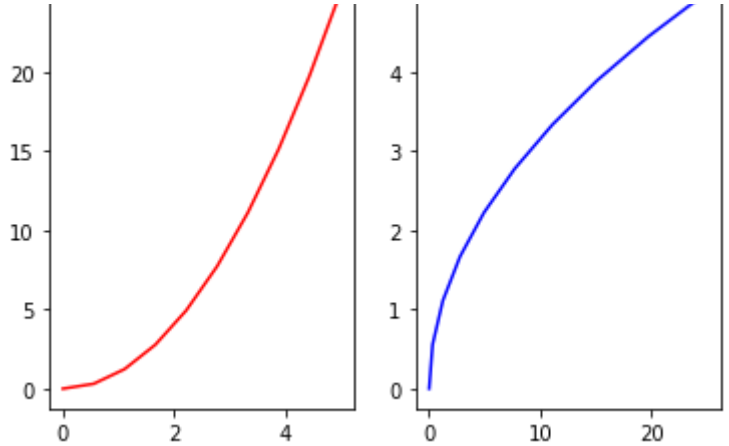
Out[404]: Text(0, 0.5, 'Days Squared')




## Print Multiple Plots

In [405]:
```
# You can print multiple plots at once
# Define the row and column to print the plot with a number assigned
# to the plot
plt.subplot(1,2,1)
# Make the line red
plt.plot(x_1,y_1,'r')

plt.subplot(1,2,2)
plt.plot(y_1,x_1,'b')
```

Out[405]: [<matplotlib.lines.Line2D at 0x7f9fd2558b90>]

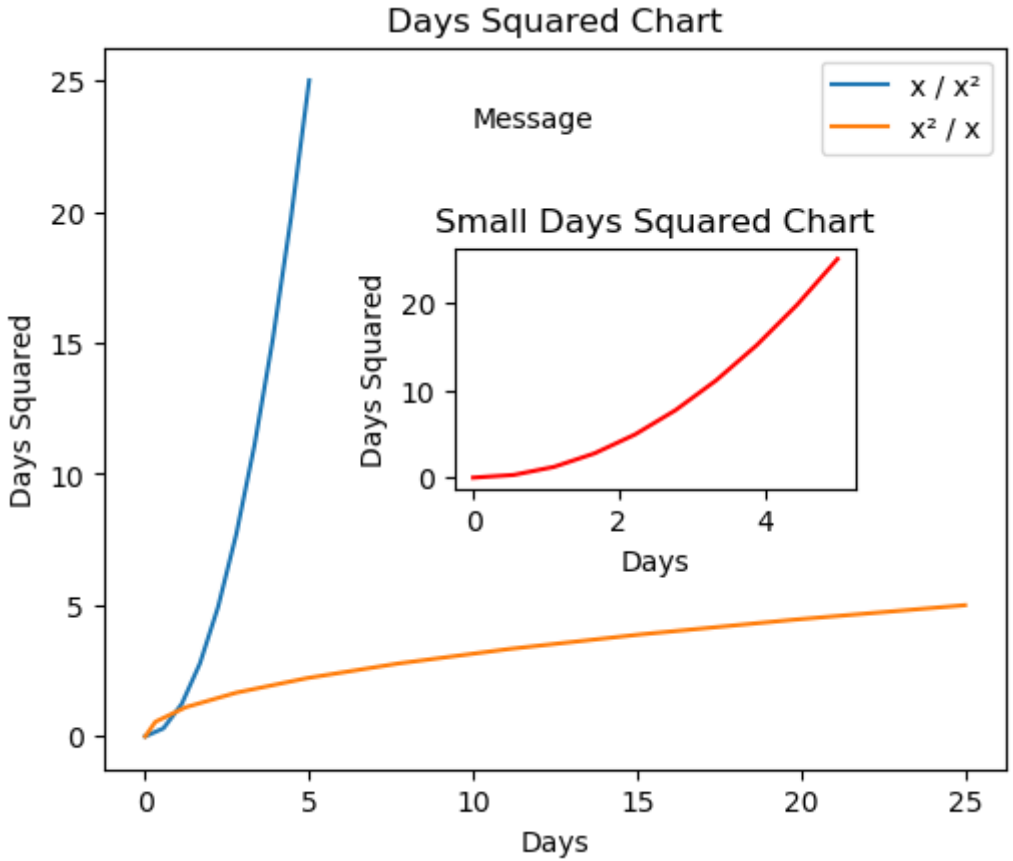



## Using Figure Objects

In [406]:
```
# A figure is an object that contains all the plot elements
# It can contain many axes
# Define width & height in inches
# Dots per inch
fig_1 = plt.figure(figsize=(5,4),dpi=100)

# Adds axes with a left, bottom, width and height that ranges from 0 to 1
# which is the percent of the canvas you want to use
axes_1 = fig_1.add_axes([0.1,0.1,0.9,0.9])

# Set lables and title
axes_1.set_xlabel('Days')
axes_1.set_ylabel('Days Squared')
axes_1.set_title('Days Squared Chart')
# Plot on the axes (If you want a label associated with the legend
# add it with label)
axes_1.plot(x_1,y_1,label='x / x²')

# You can plot to plots using the same axes
axes_1.plot(y_1,x_1,label='x² / x')

# Add the optional legend with a location number (best: 0,
# upper right: 1, upper left: 2, lower left: 3, lower right: 4,
# https://matplotlib.org/3.3.1/api/_as_gen/matplotlib.pyplot.legend.html)
```

```
# or supply a tuple of x & y from lower left
axes_1.legend(loc=0)

# You can create axis inside of others
axes_2 = fig_1.add_axes([0.45,0.45,0.4,0.3])
axes_2.set_xlabel('Days')
axes_2.set_ylabel('Days Squared')
axes_2.set_title('Small Days Squared Chart')
axes_2.plot(x_1,y_1,'r')

# Add text to plot from central point of 0,0
axes_2.text(0, 40, 'Message')
```

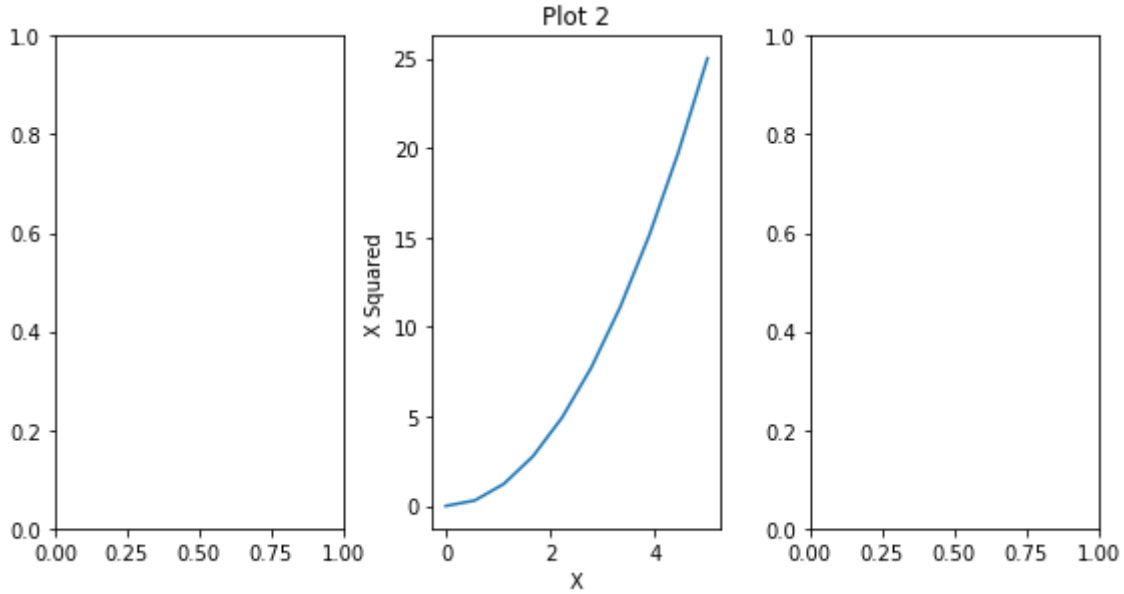
Out[406]: Text(0, 40, 'Message')



## SubPlots

In [407]:
```
# You can define multiple plots with subplots and it handles creating
# the axes objects
# axes_2 is a list of axes objects
fig_2, axes_2 = plt.subplots(figsize=(8,4), nrows=1, ncols=3)
# Put space between plots
plt.tight_layout()

# You can access the plots by index
axes_2[1].set_title('Plot 2')
axes_2[1].set_xlabel('X')
axes_2[1].set_ylabel('X Squared')
axes_2[1].plot(x_1,y_1)
```

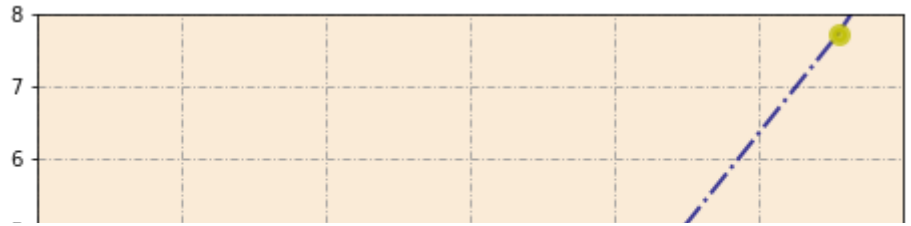
Out[407]:
```
[<matplotlib.lines.Line2D at 0x7fa01f559e10>]
```

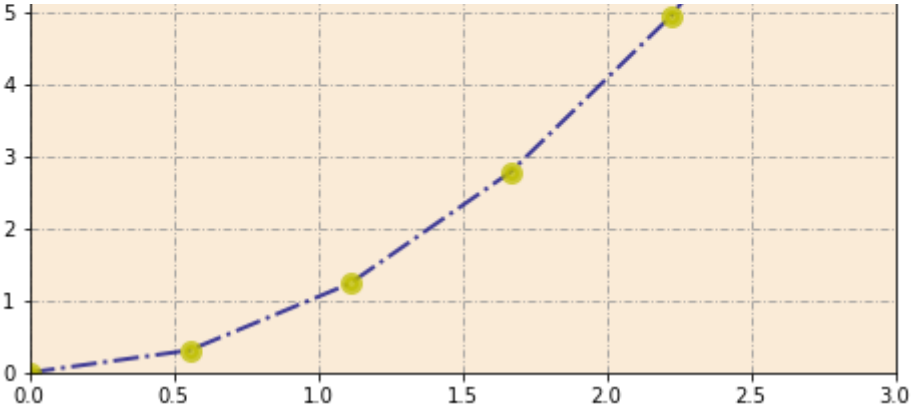



## Appearance Options

In [408]:
```
fig_3 = plt.figure(figsize=(6,4))
axes_3 = fig_3.add_axes([0,0,1,1])
```

```
# Default colors (b: blue, g: green, r: red, c: cyan, m: magenta,
# y: yellow, k: black, w: white)
# color="0.75" creates a 75% gray
# You can use hexcodes color="#eeefff"
# You can use color names found next like this color="burlywood"
# https://en.wikipedia.org/wiki/Web_colors
# alpha defines the percentage of opacity

# The default line width is 1, so to double it put in 2 and so forth

# There are many line styles
# matplotlib.org/3.1.0/gallery/lines_bars_and_markers/linestyles.html
# You can also provide a sample like '-.'

# Markers can mark your provided points on the graph
# https://matplotlib.org/3.3.0/api/markers_api.html
# You can change the markersize as well

# markerfacecolor changes the marker fill color
# markeredgecolor changes the marker stroke color
# markeredgewidth changes the markers stroke size

axes_3.plot(x_1,y_1,color='navy', alpha=.75, lw=2, ls='-.',
           marker='o', markersize=7, markerfacecolor='y',
           markeredgecolor='y', markeredgewidth=4)

# Set the lower and upper bound of x & y axis
axes_3.set_xlim([0,3])
axes_3.set_ylim([0,8])

# Add a grid, color, dashes(5pts 1 pt dashes separated by 2pt space)
axes_3.grid(True, color='0.6', dashes=(5, 2, 1, 2))
# Set grid background color
axes_3.set_facecolor('#FAEBD7')
```






## Save a Visualization to a File

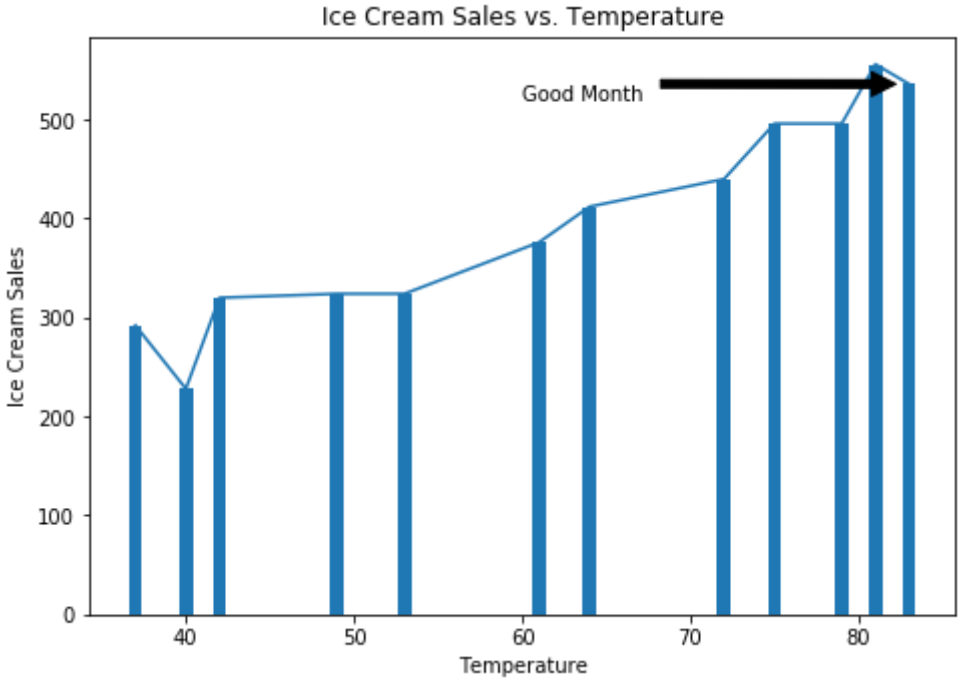
In [409]:
```
# You can save your plots to numerous file types : png, pdf, ps, eps, sv
g, pgf,
fig_3.savefig('1st_plot.png')
```

## Working with Pandas DataFrame

In [410]:
```
# Read in ice cream sales data
ics_df = pd.read_csv('icecreamsales.csv')
ics_df = ics_df.sort_values(by='Temperature')

# Convert from Pandas to NumPy array
np_arr = ics_df.values

# Get x & y values and put in array
x_2 = np_arr[:,0]
y_2 = np_arr[:,1]

fig_4 = plt.figure(figsize=(6,4))
axes_4 = fig_4.add_axes([0,0,1,1])
axes_4.set_title('Ice Cream Sales vs. Temperature')
axes_4.set_xlabel('Temperature')
axes_4.set_ylabel('Ice Cream Sales')
axes_4.plot(x_2,y_2)
```

```
# Add Annotations by supplying the x & y to point at and the position of
 the text
# based off of lower left had corner being 0,0
axes_4.annotate('Good Month', xy=(83, 536), xytext=(60, 520),
             arrowprops=dict(facecolor='black', shrink=0.05),)

# Add bars to the plot
plt.bar(x_2,y_2)
```

Out[410]: <BarContainer object of 12 artists>

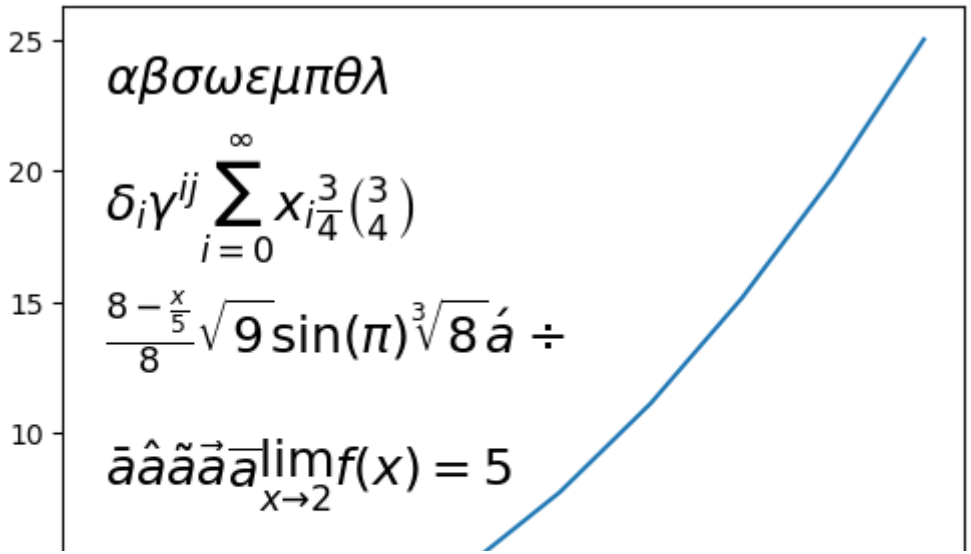


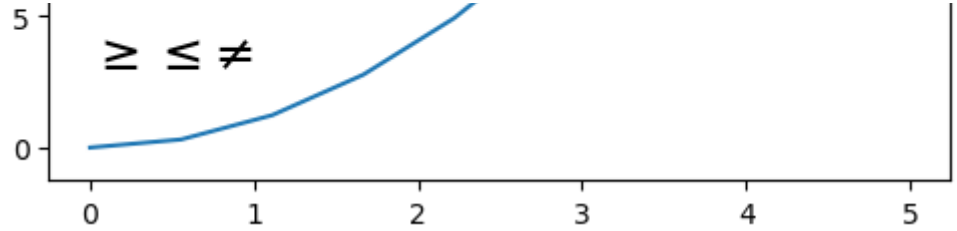

## TeX Markup

In [411]:
```
# You can use a subset of TeX markup by placing text between $
# matplotlib.org/tutorials/text/mathtext.html
fig_5 = plt.figure(figsize=(5,4),dpi=100)
axes_5 = fig_5.add_axes([0.1,0.1,0.9,0.9])

# All listed plus kappa, iota, zeta, nu, rho, eta, xi, omicron, gamma, ta
```

```
u, phi,
# chi, psi, delta (Capitalize the first letter for uppercase)
axes_5.text(0, 23,
            r'$\alpha \beta \sigma \omega \epsilon \mu \pi \theta \lambda
$', fontsize=18)
# Subscripts, multiletter superscript, sum, fractions, binomial
axes_5.text(0, 18,
            r'$\delta_i \gamma^{ij} \sum_{i=0}^\infty x_i \frac{3}{4} \bi
nom{3}{4}$',
            fontsize=18)
# Another fraction, sqrt, cbrt, trig functions :
axes_5.text(0, 13,
            r'$\frac{8 - \frac{x}{5}}{8} \sqrt{9} \sin(\pi) \sqrt[3]{8}
 \acute a \div$',
            fontsize=18)
axes_5.text(0, 8,
            r'$\bar a \hat a \tilde a \vec a \overline {a} \lim_{x \to 2}
f(x) = 5$',
            fontsize=18)
axes_5.text(0, 3,
            r'$\geq \leq \ne$',
            fontsize=18)
axes_5.plot(x_1,y_1)
```

Out[411]: [<matplotlib.lines.Line2D at 0x7f9fefa85490>]

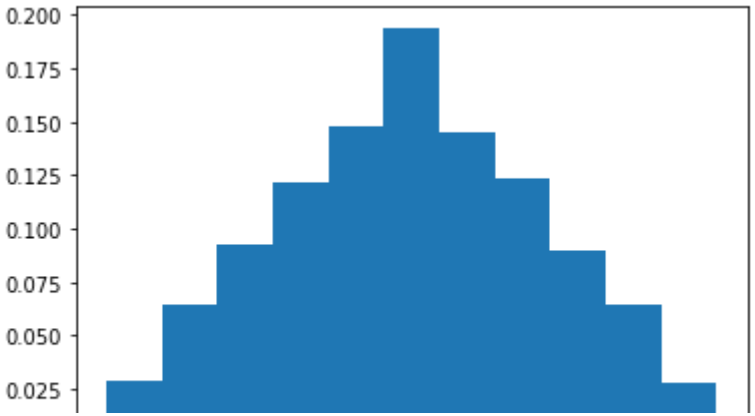





## Histograms

In [412]:
```
# Roll 2 6 sided dies get the sum and plot the histogram
arr_1 = np.random.randint(1,7,5000)
arr_2 = np.random.randint(1,7,5000)
arr_3 = arr_1 + arr_2
# Bins reprsent the number of options available 2 thru 12 = 11
# Density returns the frequency of each bin
# Range gets tuple with bin range interested in
# cumulative=True use a cumulative distribution
# histtype='step' genrates a line plot
# orientation='horizontal'
# color='orange' change bar color
plt.hist(arr_3, bins=11, density=True, stacked=True)
```

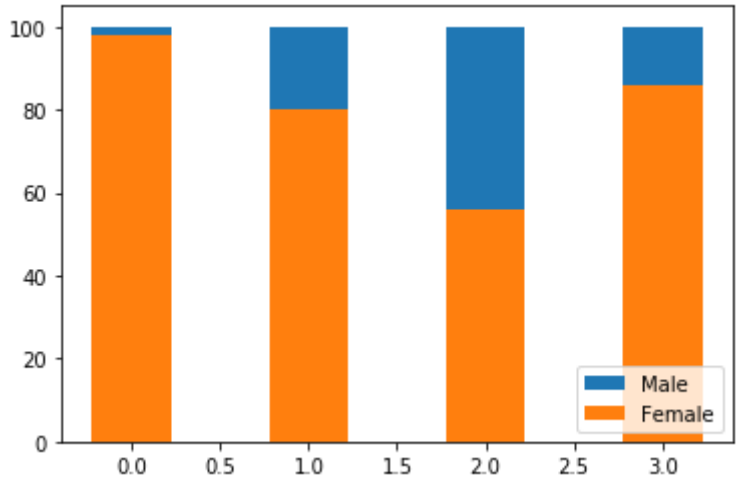
Out[412]:
```
(array([0.03, 0.06, 0.09, 0.12, 0.15, 0.19, 0.15, 0.12, 0.09, 0.06, 0.0
3]),
 array([ 2.  ,  2.91,  3.82,  4.73,  5.64,  6.55,  7.45,  8.36,  9.27,
        10.18, 11.09, 12.  ]),
 <a list of 11 Patch objects>)
```




## Bar Charts

In [413]:
```
# Analyze where France gets its electricity from
x = ['Nuclear', 'Hydro', 'Coal', 'Gas', 'Solar', 'Wind', 'Other']
per_1 = [71, 10, 3, 7, 2, 4, 3]
# Chart variance in usage
variance = [8, 3, 1, 3, 1, 2, 1]
# barh makes horizontal chart
# Also yerr, change error color with ecolor
# plt.bar(x, per_1, color='purple', yerr=variance)

# Show percentages of males & females in engineering
m_eng = (76, 85, 86, 88, 93)
f_eng = (24, 15, 14, 12, 7)

# Get evenly spaced values for each interval
spc = np.arange(5)

# Plot bars for men & women
# Can also add yerr, xerr,
# plt.bar(spc, m_eng, width=0.45, label='Male', edgecolor='k')
# plt.bar(spc + 0.45, f_eng, width=0.45, label='Female', edgecolor='k')

# Define x tick names and place in middle of bars
# plt.xticks(spc + 0.45 / 2, ('Aero', 'Chem', 'Civil', 'Elec', 'Mech'))

# Plot teachers by sex
t_type = ['Kind', 'Elem', 'Sec', 'Spec']
m_teach = np.array([2, 20, 44, 14])
f_teach = np.array([98, 80, 56, 86])
ind = [x for x, _ in enumerate(t_type)]

# Plot stacked bars for men and then women under
plt.bar(ind, m_teach, width=0.45, label='Male', bottom=f_teach)
plt.bar(ind, f_teach, width=0.45, label='Female')

plt.legend(loc='lower right')
```

Out[413]: <matplotlib.legend.Legend at 0x7f9fd2983710>

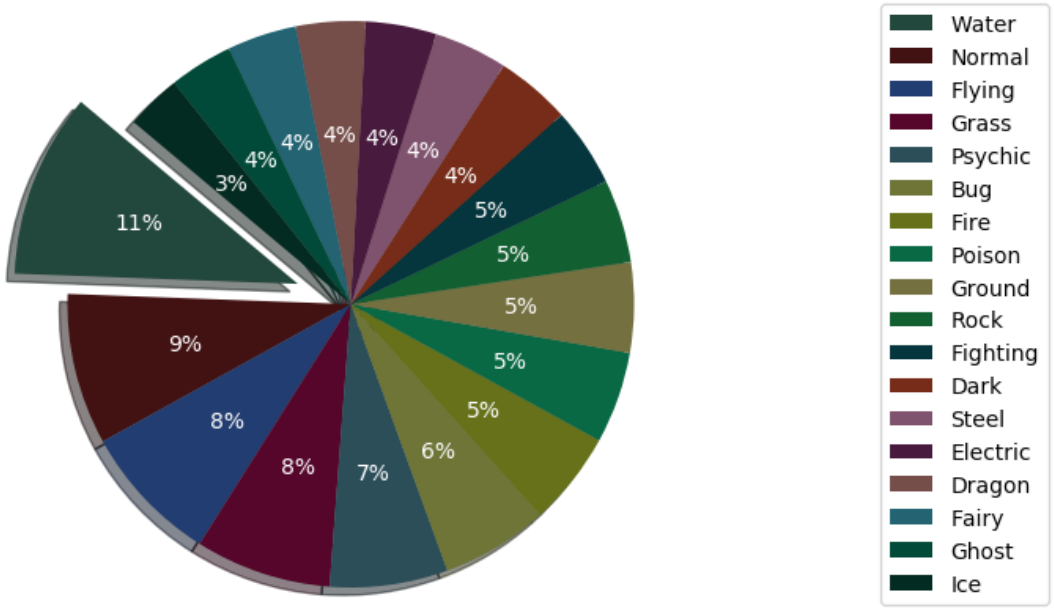



## Pie Charts

In [414]:
```
import random

fig_6 = plt.figure(figsize=(8,5),dpi=100)
axes_6 = fig_6.add_axes([0.1,0.1,0.9,0.9])

# Create a pie chart of the number of Pokemon by type
types = ['Water', 'Normal', 'Flying', 'Grass', 'Psychic', 'Bug', 'Fire',
'Poison',
'Ground', 'Rock', 'Fighting', 'Dark', 'Steel', 'Electric', 'Dragon', 'Fai
ry',
'Ghost', 'Ice']
poke_num = [133, 109, 101, 98, 85, 77, 68, 66, 65, 60, 57, 54, 53, 51, 50
, 50, 46, 40]

# Generate a random color array (Use lower values to make darkb)
colors = []
for i in range(18):
    rgb = (random.uniform(0, .5), random.uniform(0, .5), random.uniform(0
, .5))
    colors.append(rgb)
```

```
# Explode biggest 1st slice
explode = [0] * 18
explode[0] = 0.2

# Provide values, what to explode and by how much, labels, colors, pct fo
r values,
# whether to shadow, amount to rotate pie, pie text color
wedges, texts, autotexts = plt.pie(poke_num, explode=explode, labels=type
s, colors=colors,
                                   autopct='%1.0f%%', shadow=True, starta
ngle=140,
                                   textprops=dict(color="w"))
# Create legend to right and move off pie with 1-1.5 axes point width
plt.legend(wedges, types, loc='right', bbox_to_anchor=(1, 0, 0.5, 1))
```

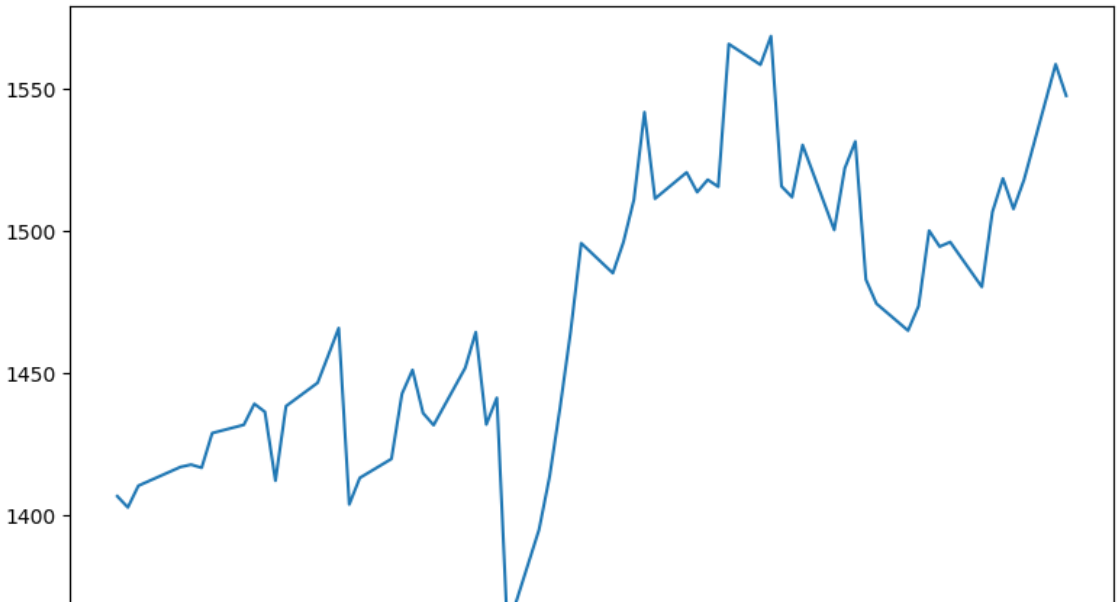
```
Out[414]: <matplotlib.legend.Legend at 0x7f9fd2ae5e10>
```

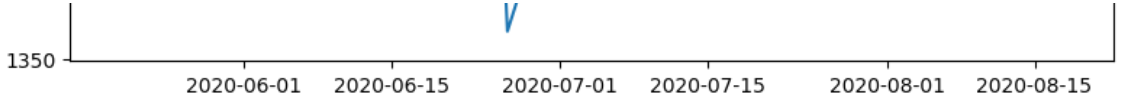



## Plot Timeseries

In [415]:
```
import datetime

# I'll show other ways of doing this when I cover Matplotlib Finance
# Get Yahoo GOOG csv file and convert to NumPy array
# https://finance.yahoo.com/quote/GOOG/history/
goog_data = pd.read_csv('GOOG.csv')
goog_data_np = goog_data.to_numpy()
# Get array of prices in 5th column
goog_cp = goog_data_np[:,4]
goog_cp

# Get NumPy array with just weekdays between dates excluding holidays
holidays = [datetime.datetime(2020,5,25), datetime.datetime(2020,8,19)]
date_arr = pd.bdate_range(start='5/20/2020', end='8/19/2020',
                          freq='C',
                         holidays=holidays)
date_arr_np = date_arr.to_numpy()

fig_7 = plt.figure(figsize=(8,5),dpi=100)
axes_7 = fig_7.add_axes([0.1,0.1,0.9,0.9])
plt.plot(date_arr_np, goog_cp)
```

Out[415]:
```
[<matplotlib.lines.Line2D at 0x7f9fd2b43d10>]
```






## Tables

In [416]:
```
# Format column data to 2 decimals
goog_data['Open'] = pd.Series([round(val, 2) for val in goog_data['Open'
]],
                              index = goog_data.index)
goog_data['High'] = pd.Series([round(val, 2) for val in goog_data['High'
]],
                              index = goog_data.index)
goog_data['Low'] = pd.Series([round(val, 2) for val in goog_data['Low']],
                              index = goog_data.index)
goog_data['Close'] = pd.Series([round(val, 2) for val in goog_data['Clos
e']],
                              index = goog_data.index)
goog_data['Adj Close'] = pd.Series([round(val, 2) for val in goog_data['A
dj Close']],
                              index = goog_data.index)

# Get most recent last 5 days of stock data
stk_data = goog_data[-5:]
stk_data

# Define headers
col_head = ('Date','Open','High','Low','Close','Adj Close','Volume')

stk_data_np = stk_data.to_numpy()
stk_data_np

# Add padding around cells in table
plt.figure(linewidth=2, tight_layout={'pad':.5}, figsize=(5,3))

# Get rid of axes and plot box
axes_8 = plt.gca()
axes_8.get_xaxis().set_visible(False)
axes_8.get_yaxis().set_visible(False)
plt.box(on=None)
```

```
# np.full returns an array filled with 0.1
# cm is a colormap object we are using to use a default blue color
# matplotlib.org/3.1.0/tutorials/colors/colormaps.html
ccolors = plt.cm.Blues(np.full(len(col_head), 0.2))

# Receives data, loc, list of column headers, column header color as arra
y of colors
# You can also add rowLabel, rowColours, rowLoc: Text alignment
the_table = plt.table(cellText=stk_data_np, loc='center', colLabels=col_h
ead,
                      colColours=ccolors)
# Set table font size
the_table.set_fontsize(14)
the_table.scale(3, 2.5)
```

```
/Users/derekbanas/opt/anaconda3/lib/python3.7/site-packages/IPython/core/
pylabtools.py:132: UserWarning: Tight layout not applied. The left and ri
ght margins cannot be made large enough to accommodate all axes decoratio
ns.
  fig.canvas.print_figure(bytes_io, **kw)
```

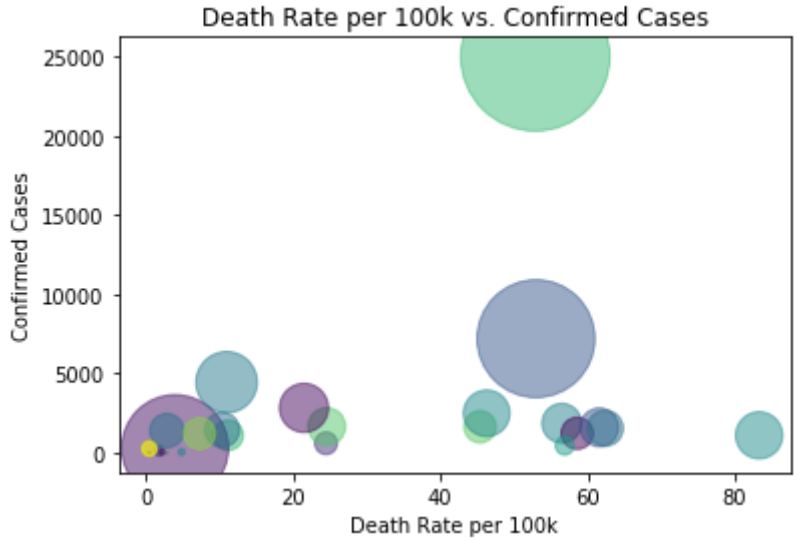
| Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| 2020-08-13 | 1510.34 | 1537.25 | 1508.01 | 1518.45 | 1518.45 | 1455200 |
| 2020-08-14 | 1515.66 | 1521.9 | 1502.88 | 1507.73 | 1507.73 | 1354800 |
| 2020-08-17 | 1514.67 | 1525.61 | 1507.97 | 1517.98 | 1517.98 | 1378300 |
| 2020-08-18 | 1526.18 | 1562.47 | 1523.71 | 1558.6 | 1558.6 | 2027100 |
| 2020-08-19 | 1553.31 | 1573.68 | 1543.95 | 1547.53 | 1547.53 | 1660000 |

## Scatterplots

In [417]:
```
# Country array
cnt_arr = np.array(['Australia','Brazil','Canada','Chile','France','Germa
ny','Greece',
                    'Iceland','India','Iran','Italy','Mexico','New Zealan
d','Nigeria',
                    'Norway','Pakistan','Peru','Russia','Saudi Arabia','Si
ngapore',
                    'South Africa','Spain','Sweden','Turkey','UK','US'])
# Death rate per 100k Coronavirus
```

```
dr_arr = np.array([1.8,53,24.5,56.5,45.4,11.2,2.2,
                   2.8,4,24.6,58.6,46.3,.5,.5,
                   4.9,2.9,83.3,11,10.4,.5,
                   21.5,61.6,56.9,7.3,62.4,52.9])
# Daily confirmed cases (Tests)
test_arr = np.array([110,7197,600,1862,1636,1103,35,
                   10,295,1658,1226,2490,8,243,
                   48,1395,1101,4447,1443,280,
                   2830,1602,447,1205,1546,24988])
# Dot size Confirmed cases
cc_arr = np.array([24236,3456652,125408,390037,256534,229706,7684,
                   2035,2836925,350279,255278,537031,1654,50488,
                   10162,290445,549321,935066,302686,56031,
                   596060,370867,85411,253108,323008,5529824])
cc_arr_sm = cc_arr / 1000
color_arr = np.random.rand(26)

plt.title('Death Rate per 100k vs. Confirmed Cases')
plt.xlabel('Death Rate per 100k')
plt.ylabel('Confirmed Cases')
plt.scatter(dr_arr,test_arr,s=cc_arr_sm,c=color_arr,alpha=0.5)
```

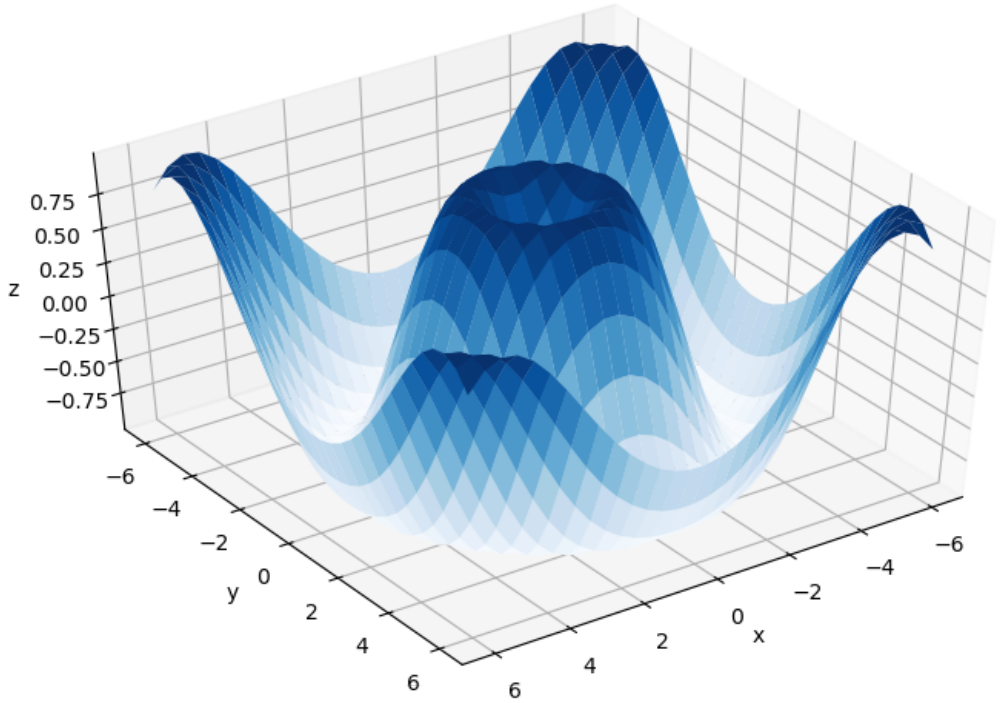
Out[417]: <matplotlib.collections.PathCollection at 0x7f9fefc34650>



## 3D Surface

In [418]:
```
# Needed for creating 3D axes
from mpl_toolkits import mplot3d

fig_9 = plt.figure(figsize=(8,5),dpi=100)
axes_9 = fig_9.add_axes([0.1,0.1,0.9,0.9], projection='3d')

# Create a 3D scatterplot
# Te darker points are represented that way to seem closer to you
z_3 = 15 * np.random.random(100)
x_3 = np.sin(z_3) * np.random.randn(100)
y_3 = np.cos(z_3) * np.random.randn(100)
# axes_9.scatter3D(x_3, y_3, z_3, c=z_3, cmap='Blues')

# You can create contour plots by defining a function for z based on x &
 y
def get_z(x, y):
    return np.sin(np.sqrt(x**2 + y**2))

x_4 = np.linspace(-6, 6, 30)
y_4 = np.linspace(-6, 6, 30)

# Creates a rectangular grid out of 2 given 1D arrays
x_4, y_4 = np.meshgrid(x_4, y_4)
z_4 = get_z(x_4,y_4)

# Change viewing angle to reorient camera 60 degrees and rotate 55
axes_9.view_init(45,55)

# Provide x, y, z, contours and color map
# axes_9.contour3D(x_4,y_4,z_4,80,cmap='Blues')
axes_9.set_xlabel('x')
axes_9.set_ylabel('y')
axes_9.set_zlabel('z')

# You can create wireframes
# axes_9.plot_wireframe(x_4,y_4,z_4,color='blue')

# You can create surface plots which is wireframe with filled faces
axes_9.plot_surface(x_4,y_4,z_4, rstride=1, cstride=1,cmap='Blues',edgeco
lor='none')
```

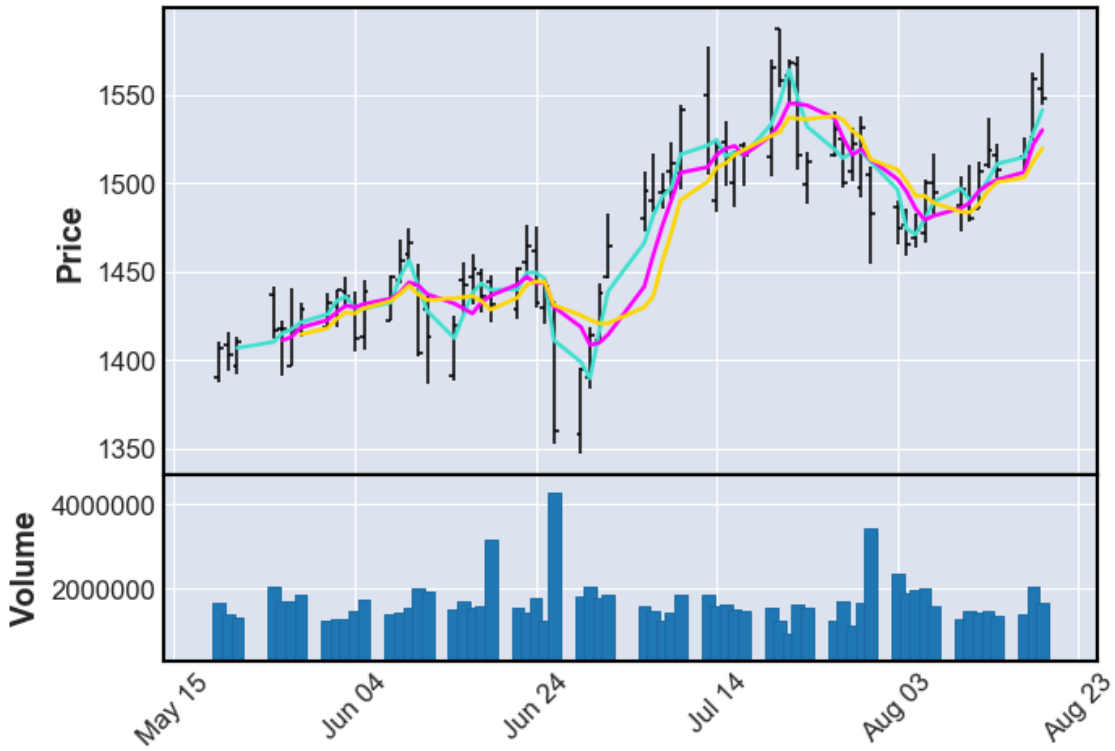
Out[418]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x7fa00743a150>




## Matplotlib Finance

In [428]:
```
# We need this module to handle the calculations
import mplfinance as mpf

# Get stock data as DataFrame and define index
goog_df = pd.read_csv('GOOG.csv',index_col=0,parse_dates=True)
goog_df.index.name = 'Date'

goog_df.shape

# A candlestick chart demonstrates the daily open, high, low and closing
 price of a stock
# mpf.plot(goog_df,type='candle')

# Plot price changes
```

```
# mpf.plot(goog_df,type='line')

# Moving averages provide trend information (Average of previous 4 observ
ations)
# mpf.plot(goog_df,type='ohlc',mav=4)

# You can plot multiple MAVs, volume, non-trading days
mpf.plot(goog_df,type='ohlc',mav=(3,5,7),volume=True,show_nontrading=True
)

# You can make additonal charts with intraday data
```




## Heatmaps

In [443]:
```
# A heatmap is a color coded representation of data from a 2D list
symptoms = ["Coronavirus","Influenza","Pneumonia","Dyspnea"]
dates = ["Jun28","Jul05","Jul12","Jul19","Jul26","Aug02","Aug09","Aug16",
```

```
"Aug21"]
symp_per = np.array([[5.2, 5.5, 5.7, 5.6, 5.3, 5.1, 5.0, 4.9, 5.3],
                     [3.5, 4.0, 4.3, 3.9, 3.5, 3.2, 2.7, 2.2, 2.0],
                     [1.8, 2.2, 2.3, 2.2, 2.1, 1.9, 1.7, 1.4, 1.3],
                     [1.0, 1.1, 1.1, 1.0, 0.9, 0.8, 0.8, 0.8, 0.7]])

fig_10, axes_10 = plt.subplots()
# Dfine data to use and color map
im = axes_10.imshow(symp_per, cmap="Wistia")

# Add ticks at data points and labels
axes_10.set_xticks(np.arange(len(dates)))
axes_10.set_yticks(np.arange(len(symptoms)))
axes_10.set_xticklabels(dates)
axes_10.set_yticklabels(symptoms)

# Rotate labels on the bottom so they don't overlap
plt.setp(axes_10.get_xticklabels(), rotation=45, ha="right",
         rotation_mode="anchor")

# Loop over data dimensions and create text annotations.
for i in range(len(symptoms)):
```