3. Introduction to **Data Vidualization**

1) df.plot(), 2) matplotlib, 3)seaborn

1. Pandas **DataFrame.plot**

In [256…
```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
```

#1 import datasets df1 and df2

In [257…
```
df1=pd.read_csv('data\\pa\\df1', index_col=0) # read the first col as index
```

In [258…
```
df1.head(2)
```

Out[258…

	A	B	C	D
2000-01-01	1.339091	-0.163643	-0.646443	1.041233
2000-01-02	-0.774984	0.137034	-0.882716	-2.253382

In [259…
```
df2=pd.read_csv('data\pa\df2')
```

In [260…
```
df2.head(2)
```

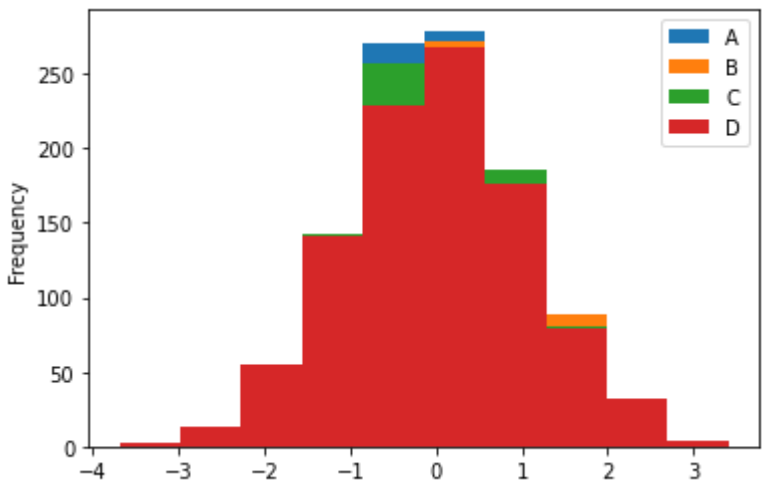
Out[260…

	a	b	c	d
0	0.039762	0.218517	0.103423	0.957904
1	0.937288	0.041567	0.899125	0.977680

#2. histogram on dataframe

In [261…
```
df1.plot.hist() #begin with df1.tab
```

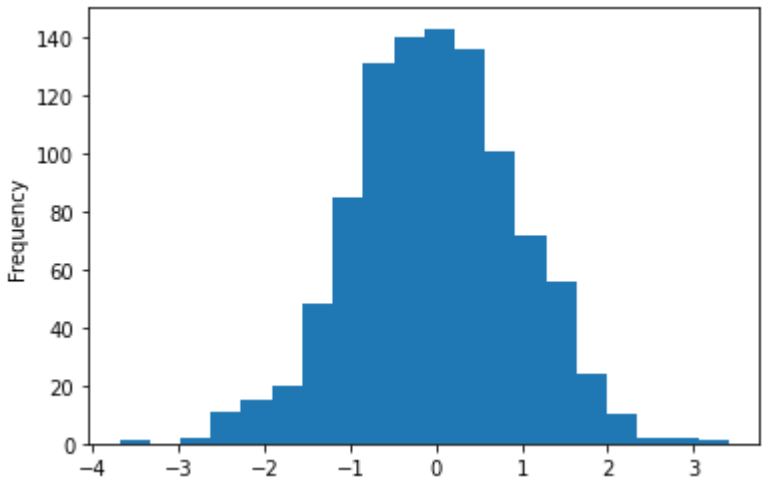
Out[261… <AxesSubplot:ylabel='Frequency'>



#3.hostogram on pd.series

In [262…
```
df1['A'].plot.hist(bins=20)
```

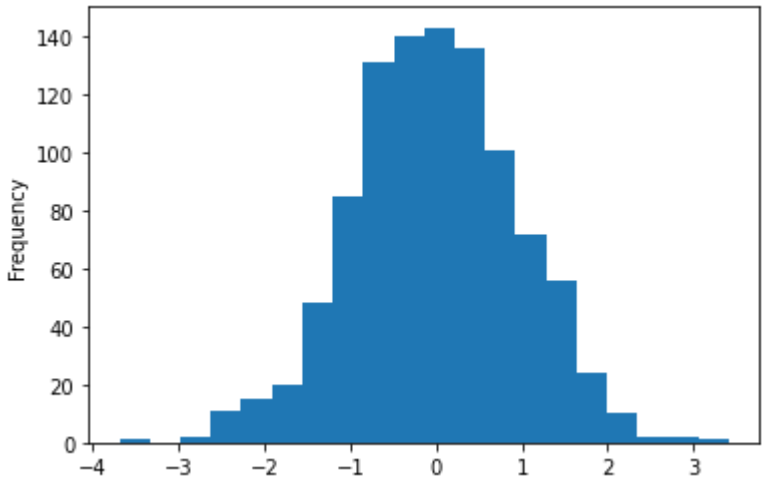
Out[262… <AxesSubplot:ylabel='Frequency'>



#4 alternative to df.plot.hist()

In [263…
```
df1['A'].plot(kind='hist', bins=20) #<-same as above
```

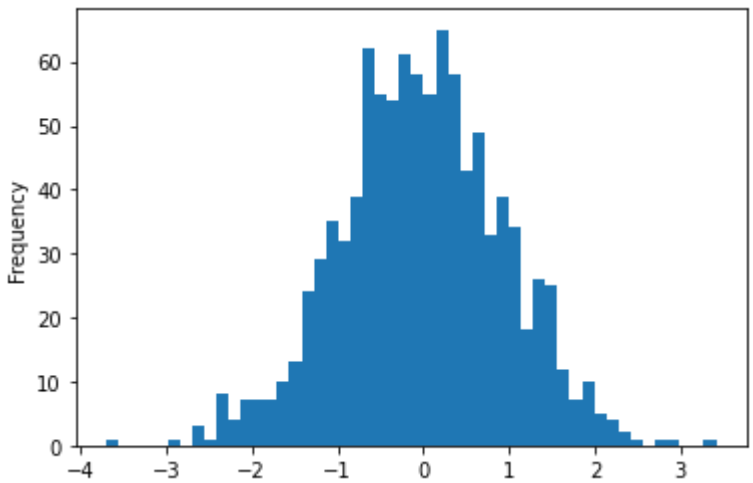
Out[263…
```
<AxesSubplot:ylabel='Frequency'>
```



#5 histogram: modify granularity by setting bins=

In [264…
```
df1['A'].plot.hist(bins=50)
```

Out[264…
```
<AxesSubplot:ylabel='Frequency'>
```



In [265…
```
df2.head(2)
```

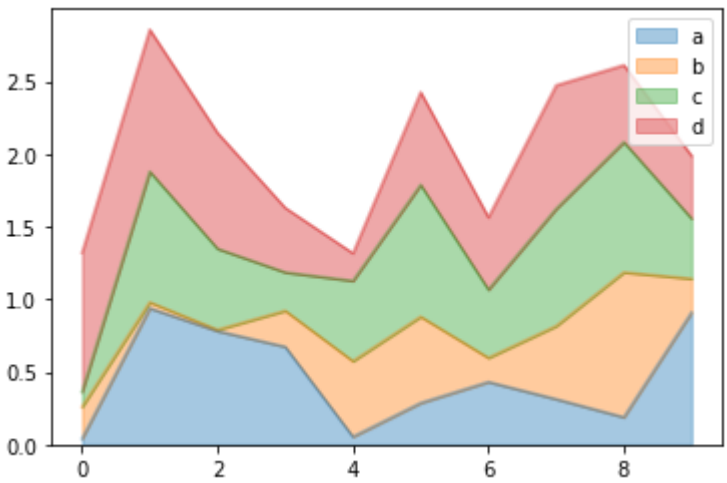
Out[265…

	a	b	c	d
0	0.039762	0.218517	0.103423	0.957904
1	0.937288	0.041567	0.899125	0.977680

#6 pd.plot.area(): and alpha for transparency between 0 and 1

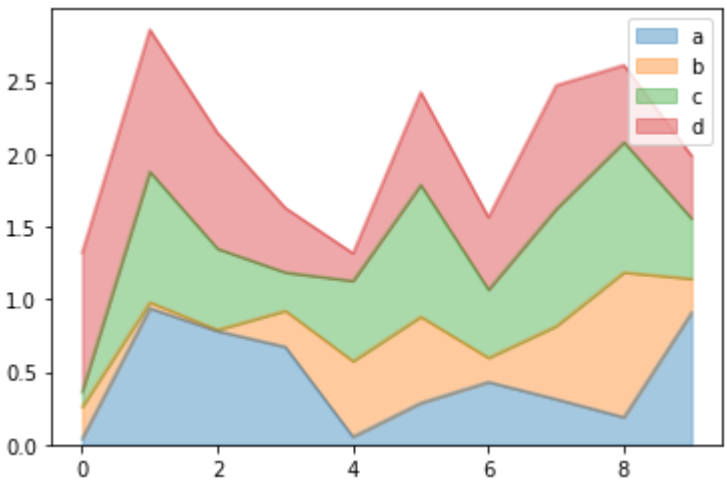
In [266…
```
df2.plot.area(alpha=.4)
```

Out[266… <AxesSubplot:>



In [267…
```
#same as above
df2.plot(kind="area", alpha=.4)
```

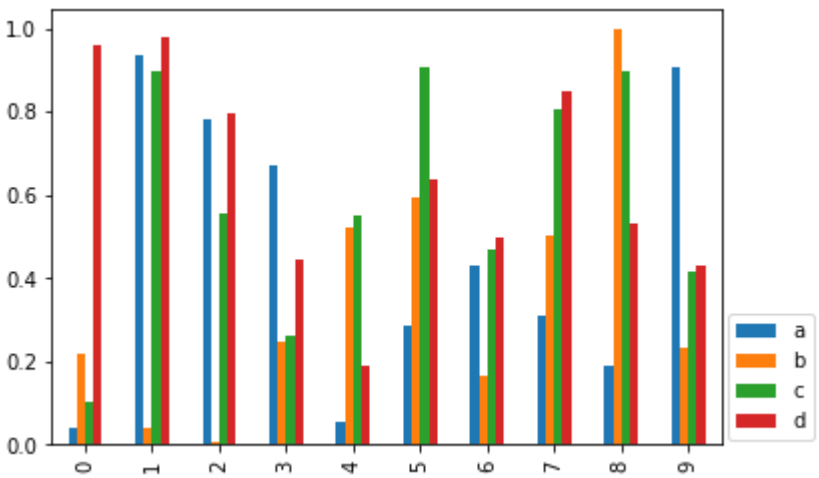
Out[267… <AxesSubplot:>



#7 axes.legend() for controlling legend for df.plot() chaining .legend()

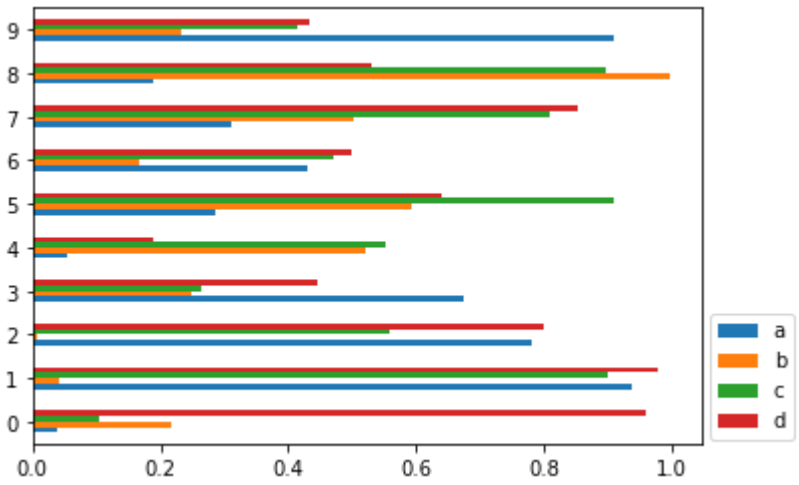
In [268…
```
ax=df2.plot.bar()
ax.legend(loc=[1.01, .01])
```

Out[268… <matplotlib.legend.Legend at 0x20e18621bb0>



In [269…
```
##2 alternative: legend control by chainning
df2.plot.barh().legend(loc=[1.01,0.01])
```

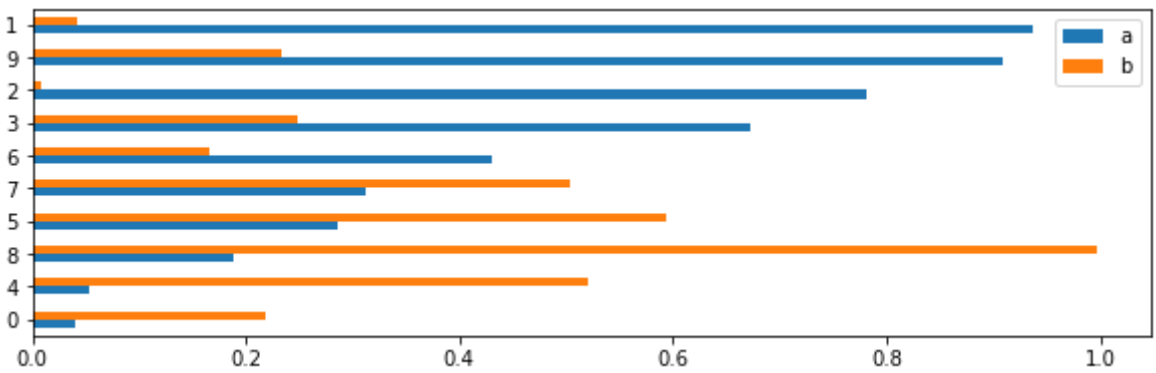
Out[269…
```
<matplotlib.legend.Legend at 0x20e1862a0d0>
```



#8 sort_values() for bar charting , figsize=()

In [270…
```
df2[['a','b']].sort_values('a').plot(kind='barh', figsize=(10,3))
```

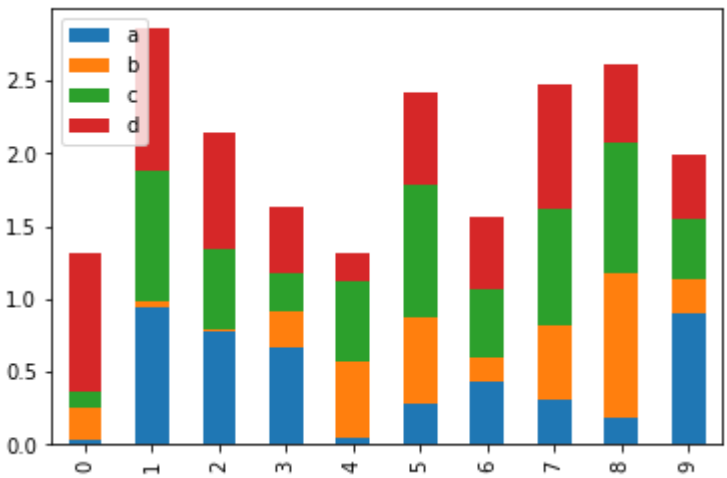
Out[270…
```
<AxesSubplot:>
```



#9 stacked=True for stacking bar chart (stacked=false is default)

In [271…
```
df2.plot.bar(stacked=True)
```

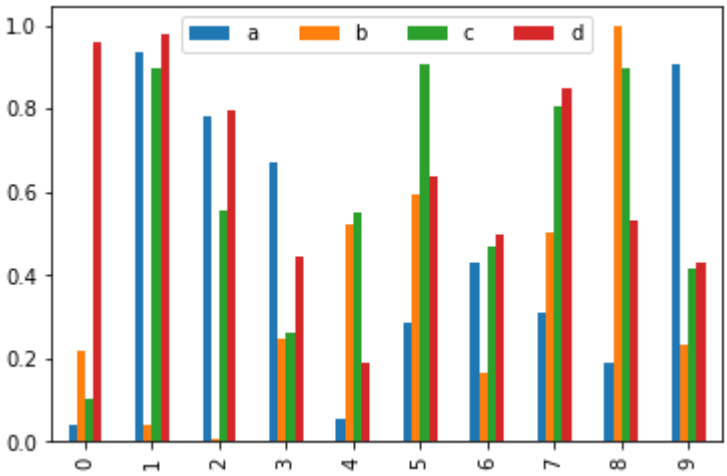
Out[271…
```
<AxesSubplot:>
```



#10 legend style controlled by ncol

In [272…
```
df2.plot.bar().legend(ncol=4)
```

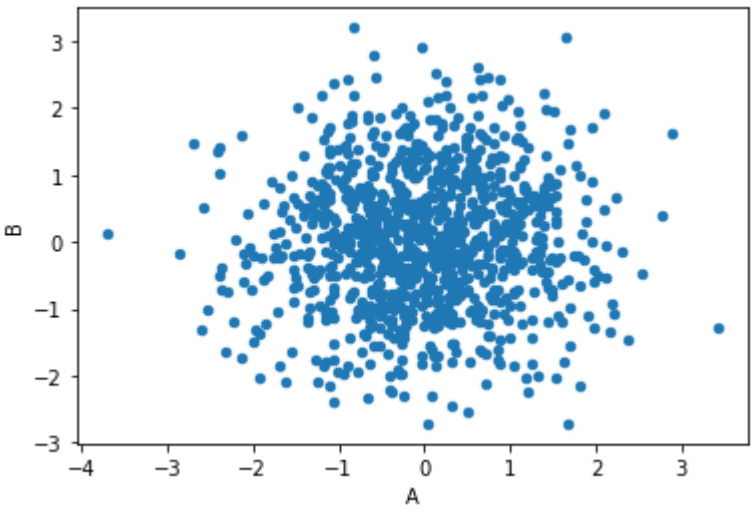
Out[272…
```
<matplotlib.legend.Legend at 0x20e18942dc0>
```



#11.1 df.plot.scatter() - caputring relationship of data points between two variables

In [273…
```
df1.plot.scatter(x='A', y='B')
```

Out[273…
```
<AxesSubplot:xlabel='A', ylabel='B'>
```



#11.2 df.plot.scatter() - introducing a 3rd variable using color

In [338…
```
#create a gender column (we arbiturarily used 'D' column as cirteria for sex +
df1['gender']=[int(a) for a in df1['D']>0]
```

In [339…
```
df1
```

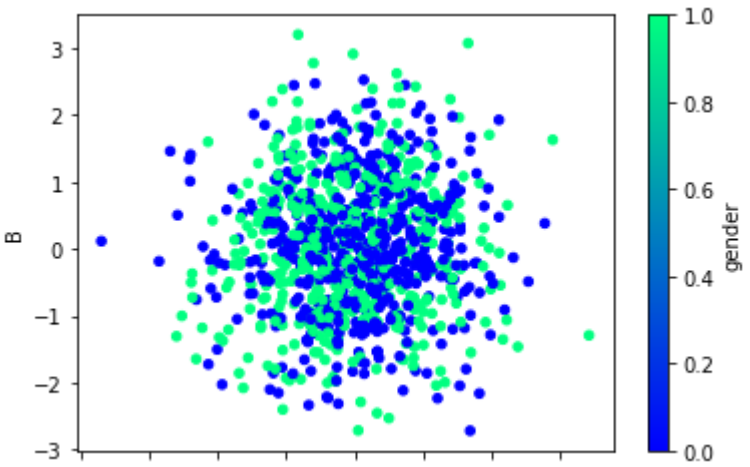
Out[339…

	A	B	C	D	gender
2000-01-01	1.339091	-0.163643	-0.646443	1.041233	1
2000-01-02	-0.774984	0.137034	-0.882716	-2.253382	0
2000-01-03	-0.921037	-0.482943	-0.417100	0.478638	1
2000-01-04	-1.738808	-0.072973	0.056517	0.015085	1
2000-01-05	-0.905980	1.778576	0.381918	0.291436	1
…	…	…	…	…	…
2002-09-22	1.013897	-0.288680	-0.342295	-0.638537	0
2002-09-23	-0.642659	-0.104725	-0.631829	-0.909483	0
2002-09-24	0.370136	0.233219	0.535897	-1.552605	0
2002-09-25	0.183339	1.285783	-1.052593	-2.565844	0
2002-09-26	0.775133	-0.850374	0.486728	-1.053427	0

1000 rows × 5 columns

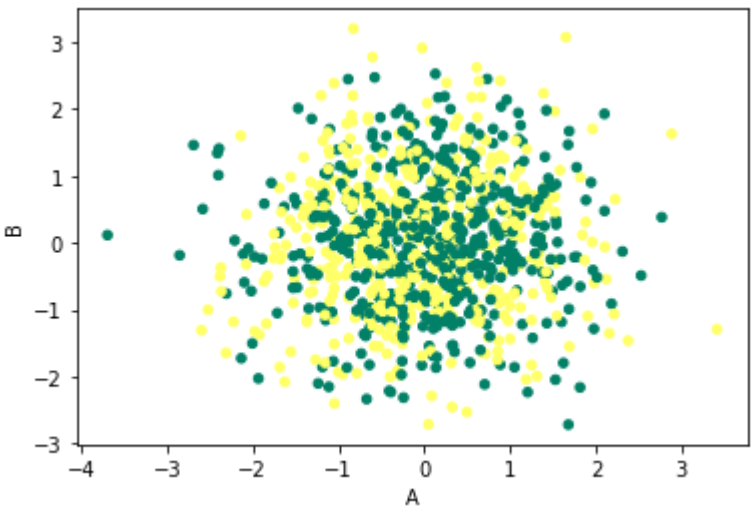
In [340…
```
df1.plot.scatter(x='A',y='B', c='gender', cmap="winter")
#<- color followed by the value of 'C' column; colorbar=False
```

Out[340…
```
<AxesSubplot:xlabel='A', ylabel='B'>
```



In [341…
```
df1.plot.scatter(x='A',y='B', c='gender', cmap='summer', colorbar=False)
#<- color followed by the value of 'C' column
```

Out[341…
```
<AxesSubplot:xlabel='A', ylabel='B'>
```



#12. df.plot.scatter() - introducing a 3rd variable using size of markers

In [342…
```
df1['score']=[c**4 for c in df1['C']]
```

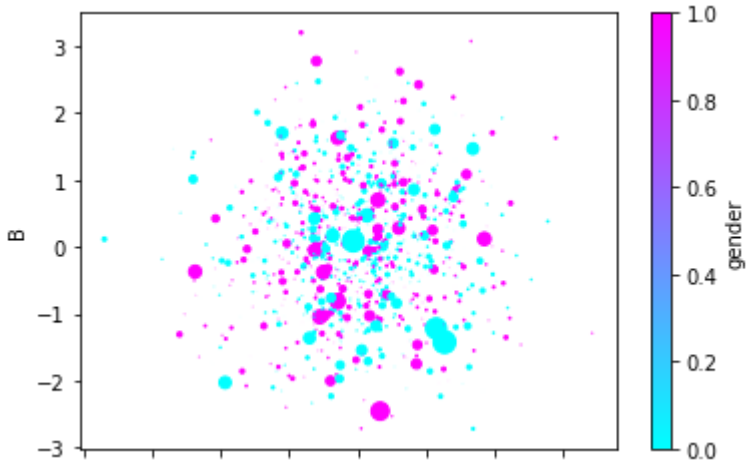
In [343…
```
df1.head(2)
```

Out[343…

	A	B	C	D	gender	score
2000-01-01	1.339091	-0.163643	-0.646443	1.041233	1	0.174630
2000-01-02	-0.774984	0.137034	-0.882716	-2.253382	0	0.607133

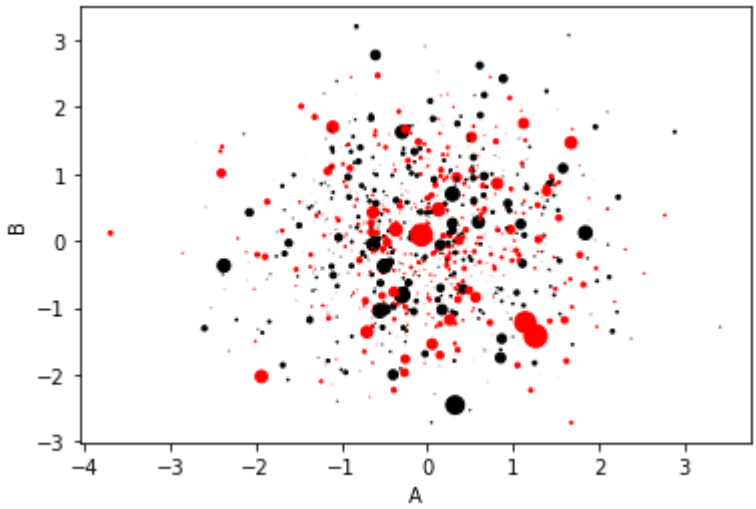
In [344…
```
df1.plot.scatter(x='A', y='B', c='gender', cmap='cool',s='score') # add shape
```

Out[344…
```
<AxesSubplot:xlabel='A', ylabel='B'>
```



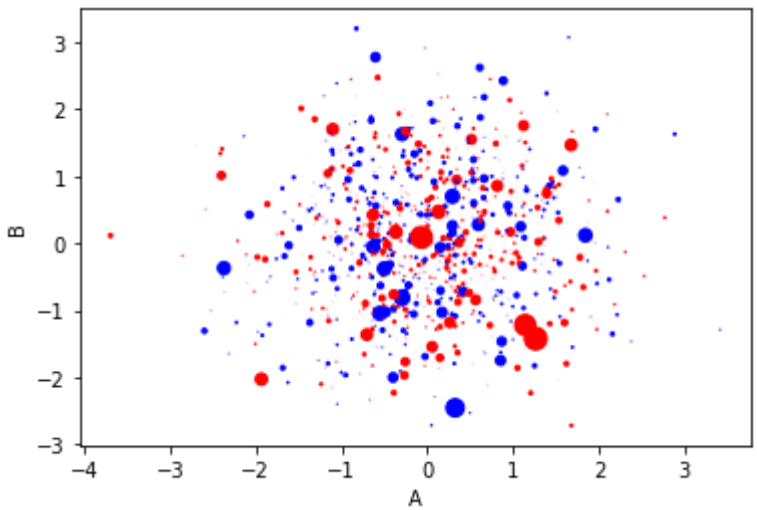
In [387…
```
mycolor={0:'red', 1:'black'}
df1.plot.scatter(x='A', y='B', c=df1['gender'].map(mycolor), s='score')
```

Out[387… <AxesSubplot:xlabel='A', ylabel='B'>



In [385…
```
mycolor={0:'red', 1:'blue'}
df1.plot.scatter(x='A', y='B', c=df1['gender'].map(mycolor), s='score')
```

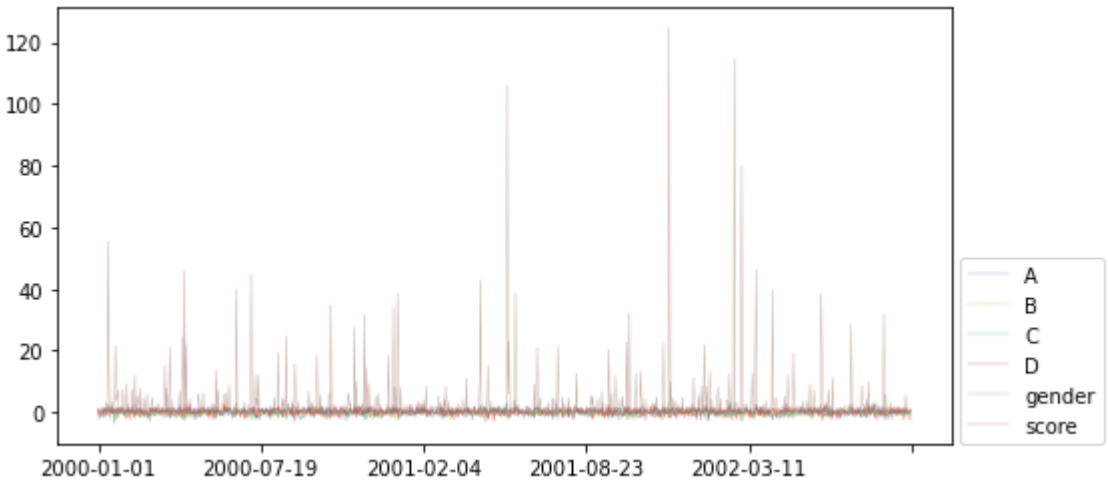
Out[385… <AxesSubplot:xlabel='A', ylabel='B'>



#13.1 df.plot.line() - line chart, figsize

In [283…
```
df1.plot.line(figsize=(8,4), lw=0.4, alpha=0.5).legend(loc=[1.01,0])
```

Out[283… <matplotlib.legend.Legend at 0x20e18cdbee0>



In [284…
```
df1['cum1'], df1['cum2']=df1['A'].cumsum(), df1['B'].cumsum()
```

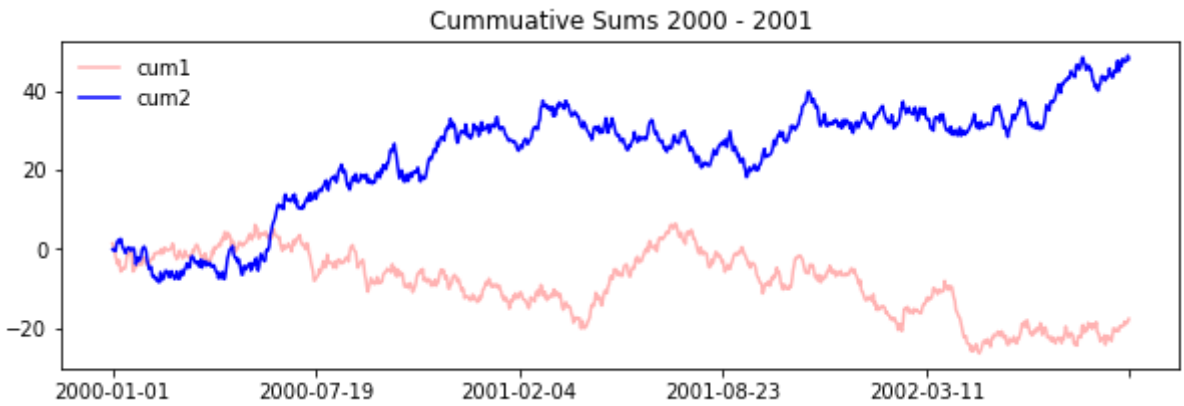
In [285…
```
df1.head(3)
```

Out[285…

	A	B	C	D	gender	score	cum1	cum2
2000-01-01	1.339091	-0.163643	-0.646443	1.041233	1	0.174630	1.339091	-0.163643
2000-01-02	-0.774984	0.137034	-0.882716	-2.253382	0	0.607133	0.564107	-0.026609
2000-01-03	-0.921037	-0.482943	-0.417100	0.478638	1	0.030266	-0.356930	-0.509552

In [286…
```
df1['cum1'].plot.line(color='r', alpha=0.3, figsize=(10,3))
df1['cum2'].plot.line(color='b')
plt.legend(frameon=False)
plt.title('Cummuative Sums 2000 - 2001')
```

Out[286…
```
Text(0.5, 1.0, 'Cummuative Sums 2000 - 2001')
```



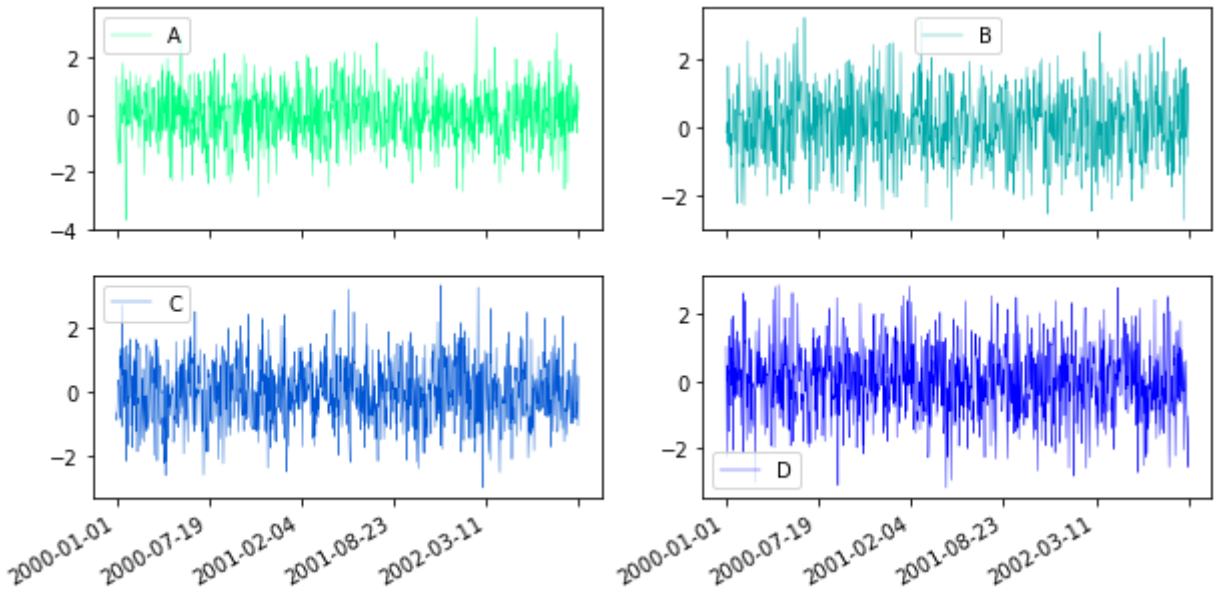
In [287…
```
df1['cum1'].plot.line(color='r', alpha=0.3, figsize=(10,3))
```

Out[287…
```
<AxesSubplot:>
```

#13.2 subplots = True

```
In [288…]  df1.iloc[:,:4].plot.line(subplots=True, lw=0.4, figsize=(10,5), cmap='winter_r
```

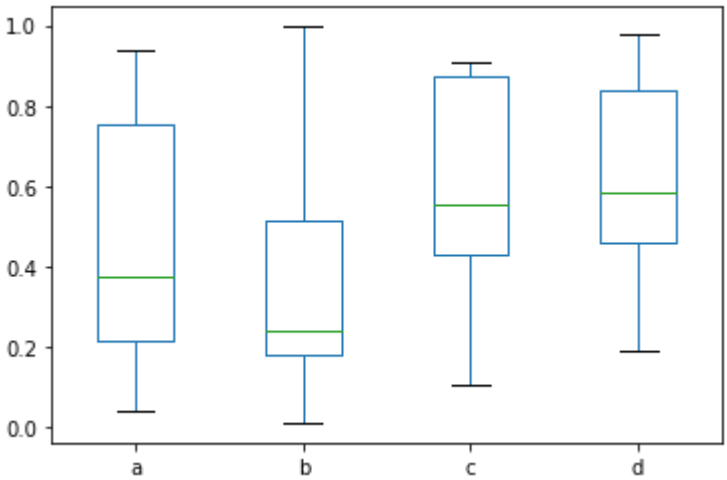
```
Out[288…]  array([[<AxesSubplot:>, <AxesSubplot:>],
                  [<AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



#14 df.plot.box() on a dataframe

```
In [289…]  df2.plot.box()
```

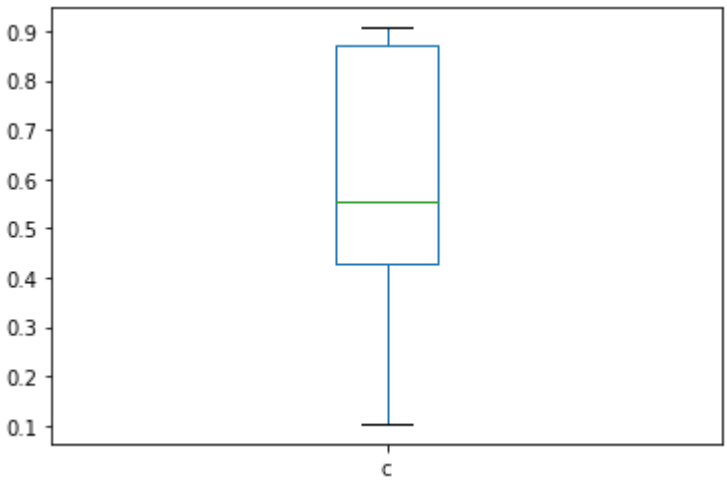
```
Out[289…]  <AxesSubplot:>
```



#15 df.plot.box() on a series

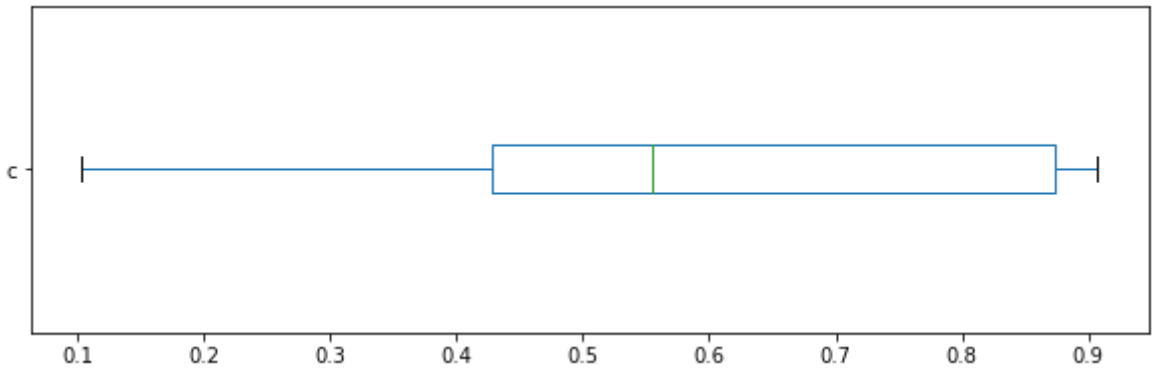
```
In [290…]  df2['c'].plot.box()
```

```
Out[290…]  <AxesSubplot:>
```



In [291…
```
df2['c'].plot.box(vert=False,figsize=(10,3)) #vert only works for box plot
```

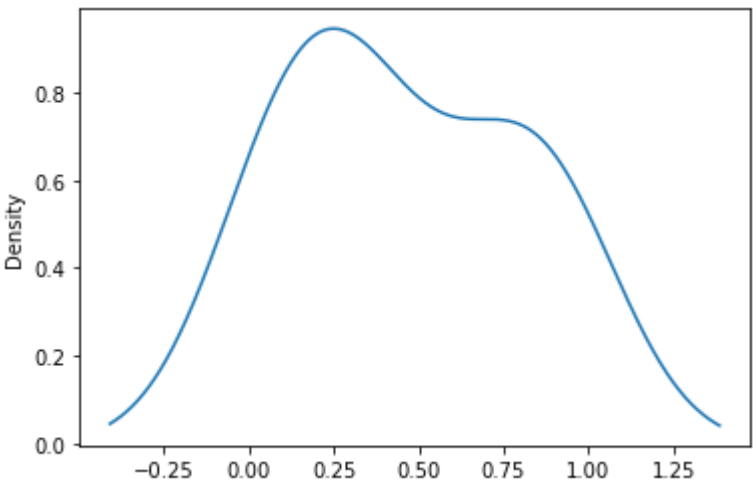
Out[291…
```
<AxesSubplot:>
```



#16 df.plot.kde() on Series kernel density estimation (KDE) - the prob integrates to 1 not sum to 1

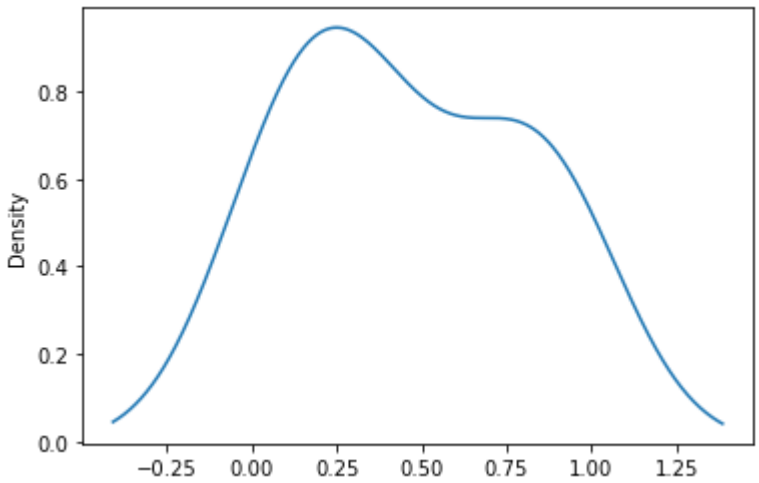
In [292…
```
df2['a'].plot.kde()
```

Out[292…
```
<AxesSubplot:ylabel='Density'>
```



In [293…
```
df2['a'].plot.density() #same as above
```

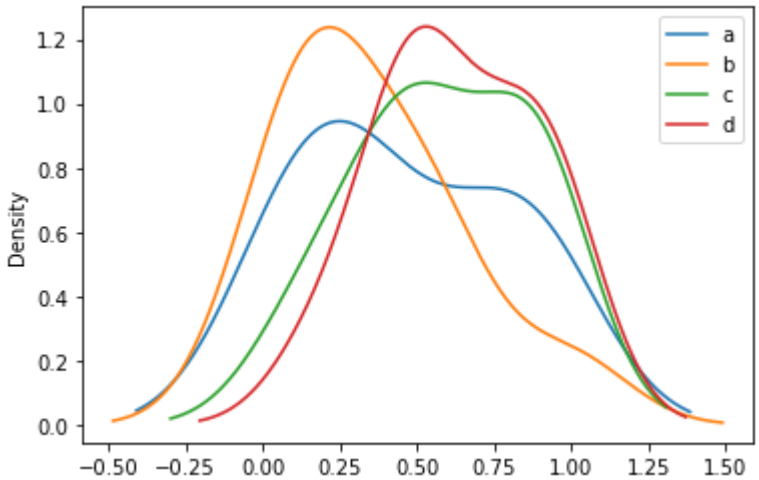
Out[293…
```
<AxesSubplot:ylabel='Density'>
```



#17 kde() on a DF

```
In [294…  df2.plot.kde() #show multiple from a dataframe
```

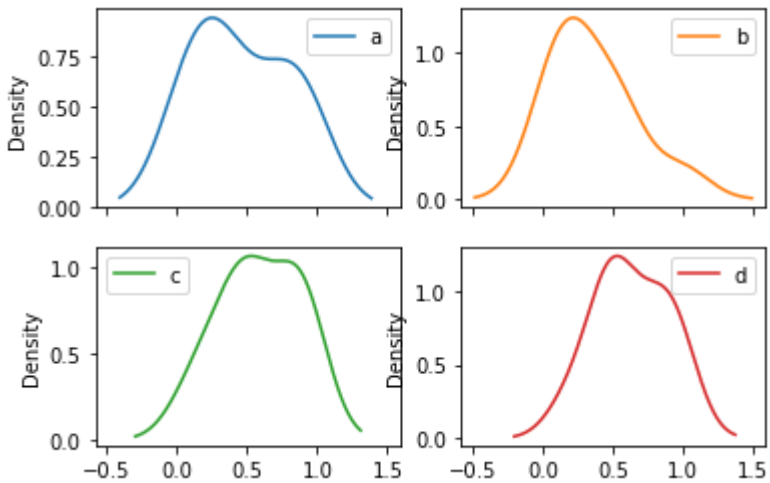
```
Out[294…  <AxesSubplot:ylabel='Density'>
```



#18 df.plot.kde() subplots and layout

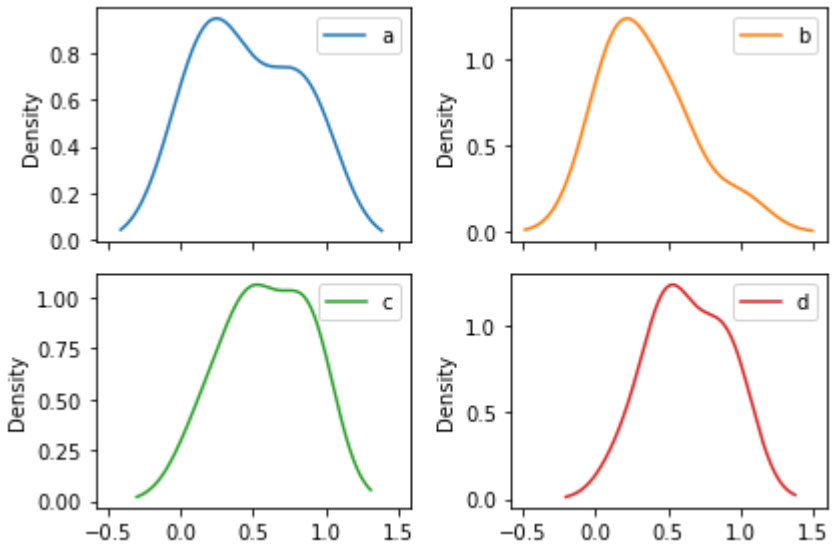
```
In [295…  df2.plot.kde(subplots=True, layout=(2,2)) #show multiple from a dataframe using
```

```
Out[295…  array([[<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
          [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>]],
         dtype=object)
```



#19 df.plot.kde()- plt.tight_layout()

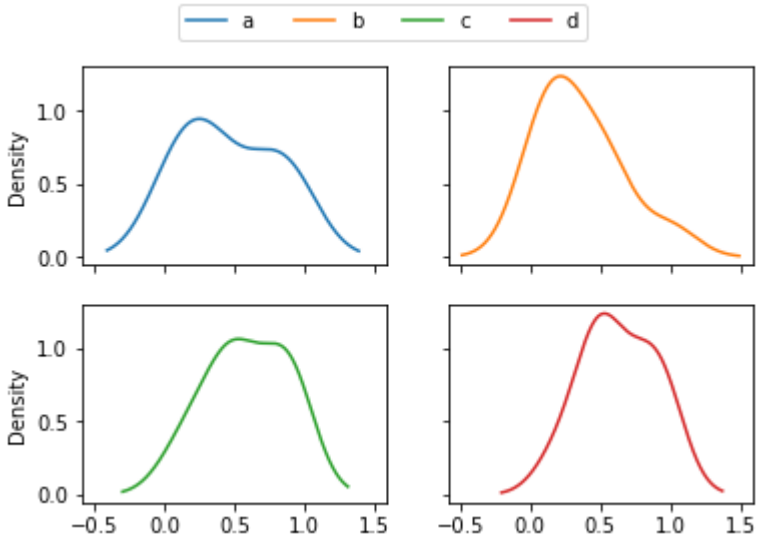
```
In [296…  df2.plot.kde(subplots=True, layout=(2,2)) #add tight_layout()
          plt.tight_layout()
```



#20 figure level legend on subplots

In [297…
```
df2.plot.kde(subplots=True, sharey=True, layout=(2,2), legend=0) # remove indi
plt.figlegend(loc='upper center', ncol=4) # create a figure level legend
```

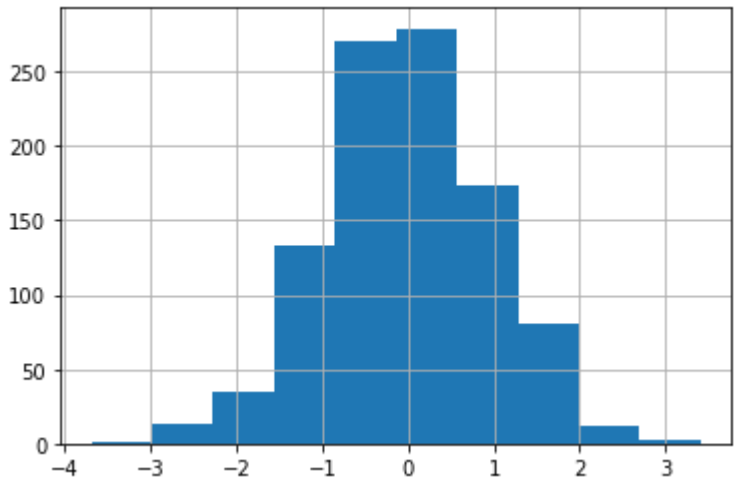
Out[297…
```
<matplotlib.legend.Legend at 0x20e1a3fa790>
```



#21 grid=False

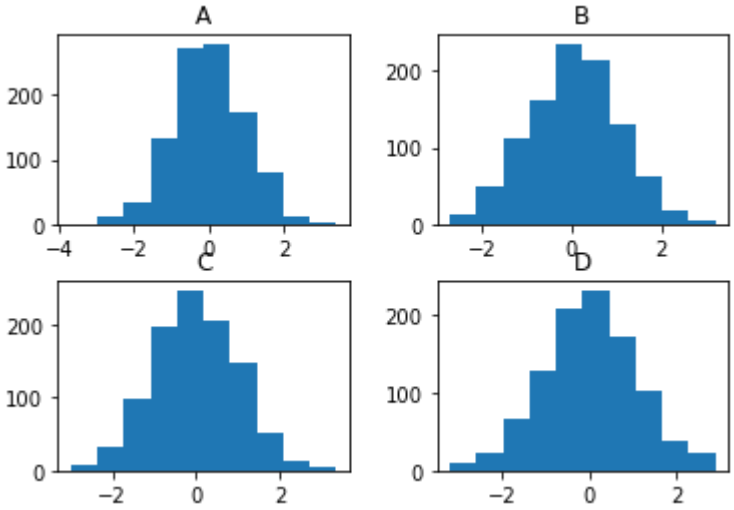
In [298…
```
df1=pd.read_csv('data\\pa\\df1', index_col=0) # read the first col as index
df1['A'].hist()  # you can use grid false to remove the background grid
```

Out[298…
```
<AxesSubplot:>
```

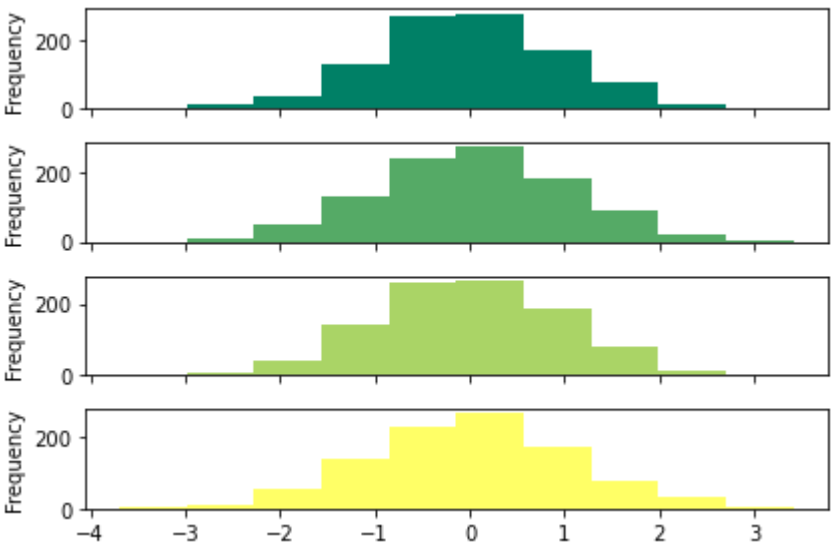


In [299…
```
df1.hist(grid=False)
```

Out[299…
```
array([[<AxesSubplot:title={'center':'A'}>,
        <AxesSubplot:title={'center':'B'}>],
       [<AxesSubplot:title={'center':'C'}>,
        <AxesSubplot:title={'center':'D'}>]], dtype=object)
```



In [300…
```
df1.plot(kind='hist', subplots=True, layout=(4,1), legend=False, cmap='summer')
plt.tight_layout()
```



Q1. Download the following data(') and save it as weather. Following the instruction below, reproduce the image examples as closely as you possible. In this assignment, you have to use only pandas dataframe object methods, not yet the matplotlib package. 1) State the number of records and columns in the weather dataset. 2) Draw a barchart that compares the average temperature of each months (hint: use groupby()) 3) Draw a barchart that compares the average temperature and humid of each months by modifying your code for 2). 4) Draw a scattr plot that shows the relationship between temperature and humidity. Control the numbers in many parameters (s= , alpha=, c=, cmap =) so you can get as closely as possible to the image. Size (s) and color(c) of the bubbles needs to be proportionally changed by the level of 'precip'. you may want to choose any cmap you like.

In [301…
```
import pandas as pd
from IPython.display import Image
```

In [302…
```
weather=pd.read_csv('https://raw.githubusercontent.com/kjmobile/data/main/airli
# or use this code weather=pd.read_csv('data\\airlines\\weather_r.csv') AFTER
```

In [303…
```
# 1) State the shape of weather data.
weather.shape
```

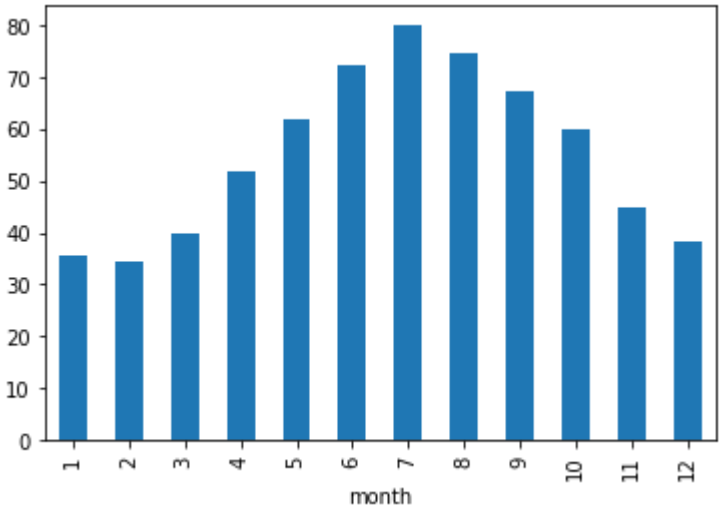
Out[303…
```
(26115, 15)
```

In [304…
```
weather.columns
```

Out[304…
```
Index(['origin', 'year', 'month', 'day', 'hour', 'temp', 'dewp', 'humid',
       'wind_dir', 'wind_speed', 'wind_gust', 'precip', 'pressure', 'visib',
       'time_hour'],
      dtype='object')
```

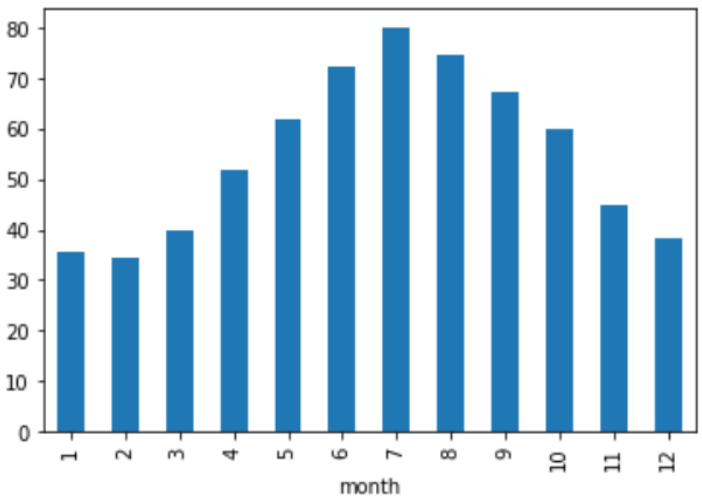
In [305…
```
# 2) Draw a barchart that compares the average temperature of each months.
weather.groupby('month').mean()['temp'].plot.bar()
```

Out[305…
```
<AxesSubplot:xlabel='month'>
```



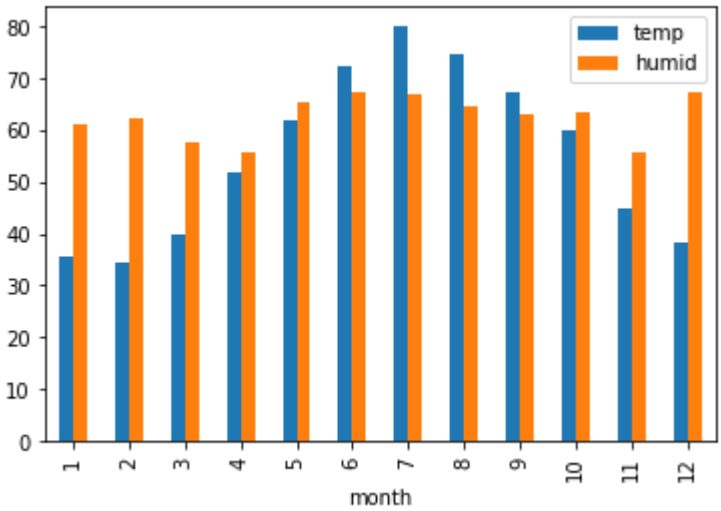
In [306…
```
Image('https://raw.githubusercontent.com/kjmobile/data/main/img/weather_bar_mo
```

Out[306…
```
<AxesSubplot:xlabel='month'>
```



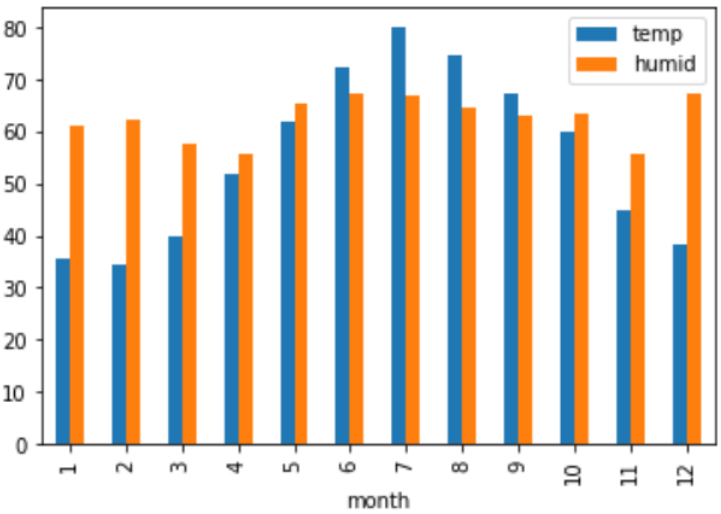
In [307…
```
# 3) Draw a barchart that compares the average temperature and humid of each m
weather.groupby('month').mean()[['temp','humid']].plot.bar()
```

Out[307…
```
<AxesSubplot:xlabel='month'>
```



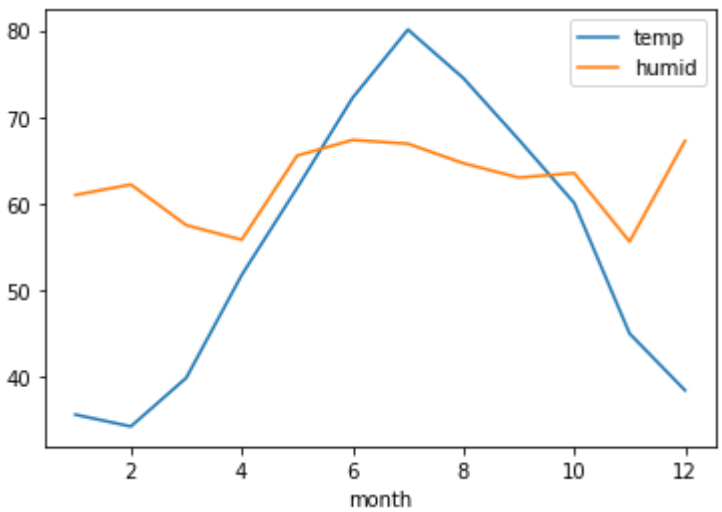
In [308…
```
Image('https://raw.githubusercontent.com/kjmobile/data/main/img/weather_bar_ter
```

Out[308…



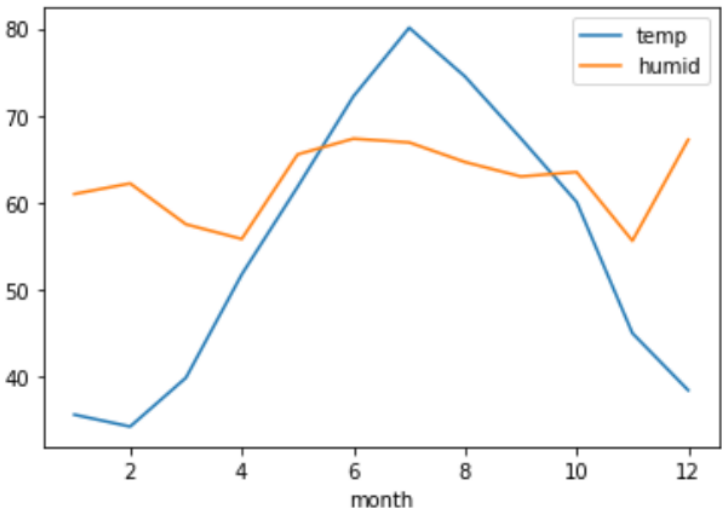
In [309…
```
# 4) Draw a line chart that compares the average temperature and humid of each
weather.groupby('month').mean()[['temp','humid']].plot.line()
```

Out[309… <AxesSubplot:xlabel='month'>



In [310…
```
Image('https://raw.githubusercontent.com/kjmobile/data/main/img/weather_line_te
```

Out[310…

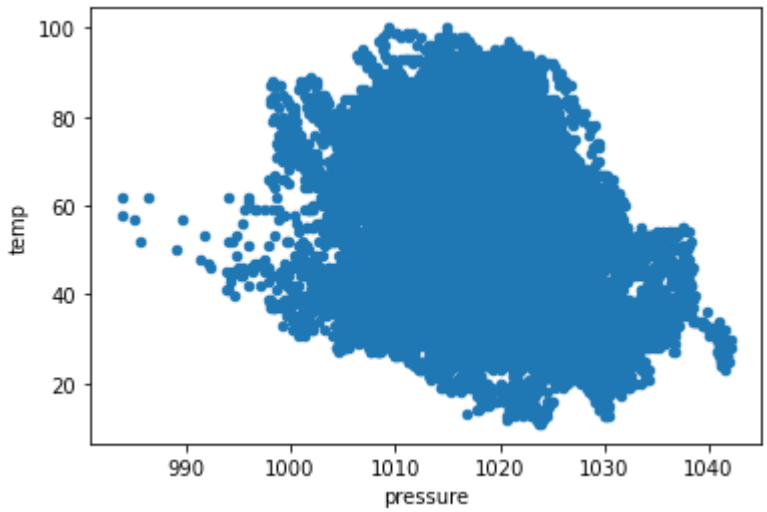


In [311…
```
weather.columns
```

Out[311…
```
Index(['origin', 'year', 'month', 'day', 'hour', 'temp', 'dewp', 'humid',
       'wind_dir', 'wind_speed', 'wind_gust', 'precip', 'pressure', 'visib',
       'time_hour'],
      dtype='object')
```

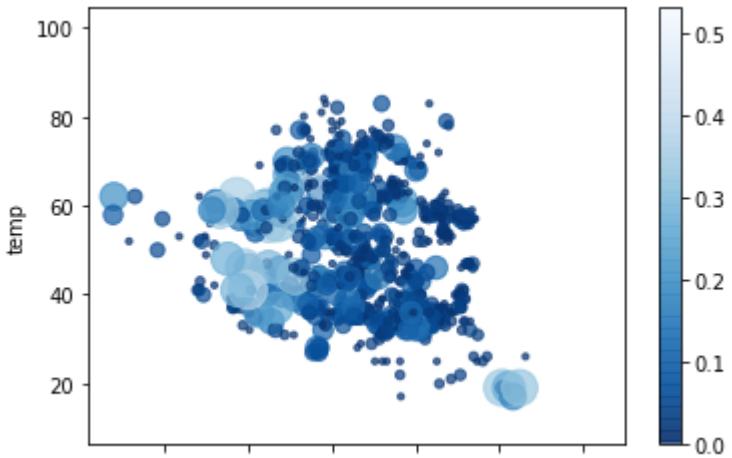
In [312…
```
weather.plot.scatter(x='pressure', y='temp')
```

Out[312…
```
<AxesSubplot:xlabel='pressure', ylabel='temp'>
```



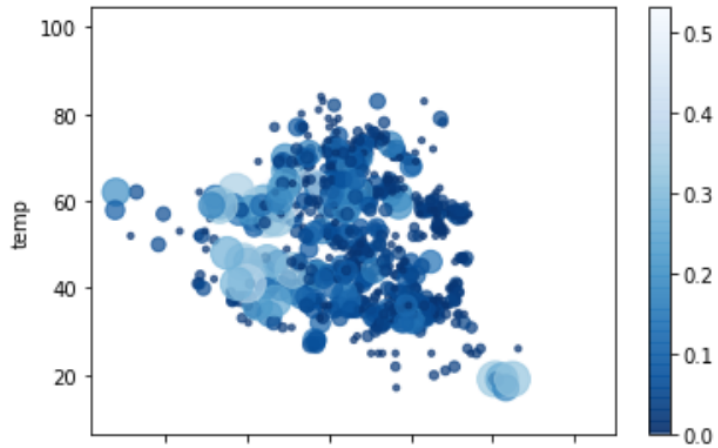
In [313…
```
# 4) Draw a scattr plot that shows the relationship between temperature and hu
#    control the numbers in many parameters ( s= , alpha=, c=, cmap = ) so you
#    size (s) and color(c) of the bubbles needs to be proportionally changed b
weather.plot.scatter(x='pressure',y='temp', s=weather['precip']*1000, alpha=0.
```

Out[313…
```
<AxesSubplot:xlabel='pressure', ylabel='temp'>
```



In [314…
```
Image('https://raw.githubusercontent.com/kjmobile/data/main/img/weather_scatte
```

Out[314…



Q2. Download the following data(country_wise_latest.csv) and save it as covid. Following the instruction below, reproduce the image examples as closely as you possible. In this assignment, you have to use only pandas dataframe object methods, not yet the matplotlib package. 1) State the shape of covid data. 2) Draw a horizontal bar chart that compares the number of deaths ('Deaths') of countries. In the chart, include only the countries that have more than 5000 deaths. Don't forget to sort the chart from large to small. 3) For the same countries as 2), draw a horizontal bar chart the compares the 'Deaths / 100 Cases'. Now US is not number one! 4) Draw a barchart that compares 'Confirmed last week' numbers by country: include only top 50 countries in number of confirmed last week. 5) Draw a single variable boxplot showing the distribution of 'Deaths / 100 Cases' in the entire dataset. 6) Draw of kernel density estimation chart showing the distribution of 'Deaths / 100 Cases' in the entire dataset.

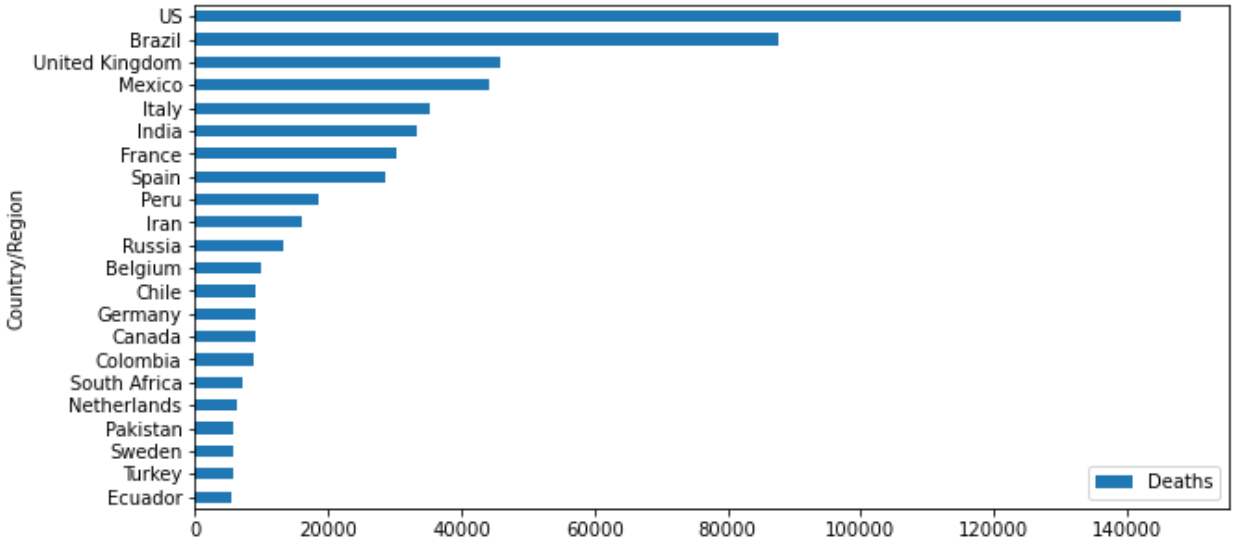
In [315…
```
covid=pd.read_csv("https://raw.githubusercontent.com/kjmobile/data/main/pa/cou
# or run this code covid = pd.read_csv('data\pa\country_wise_latest.csv') afte
```

In [316…
```
# 1)
covid.shape
```

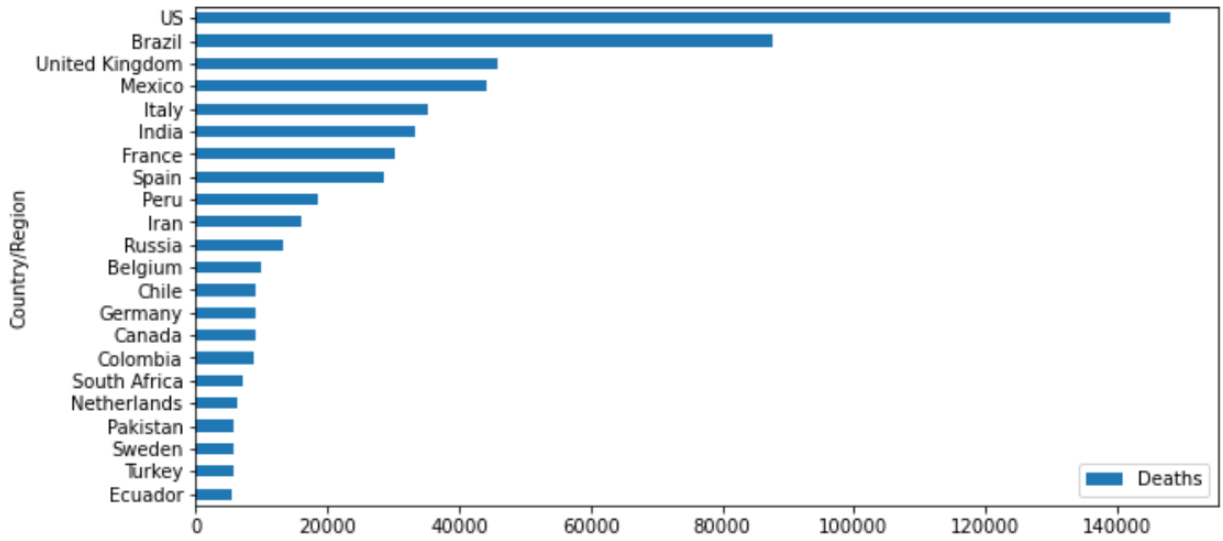
Out[316… `(187, 15)`

In [317…
```
# 2)
covid[covid['Deaths']>5000].sort_values('Deaths').plot.barh(x='Country/Region',
```

Out[317… `<AxesSubplot:ylabel='Country/Region'>`

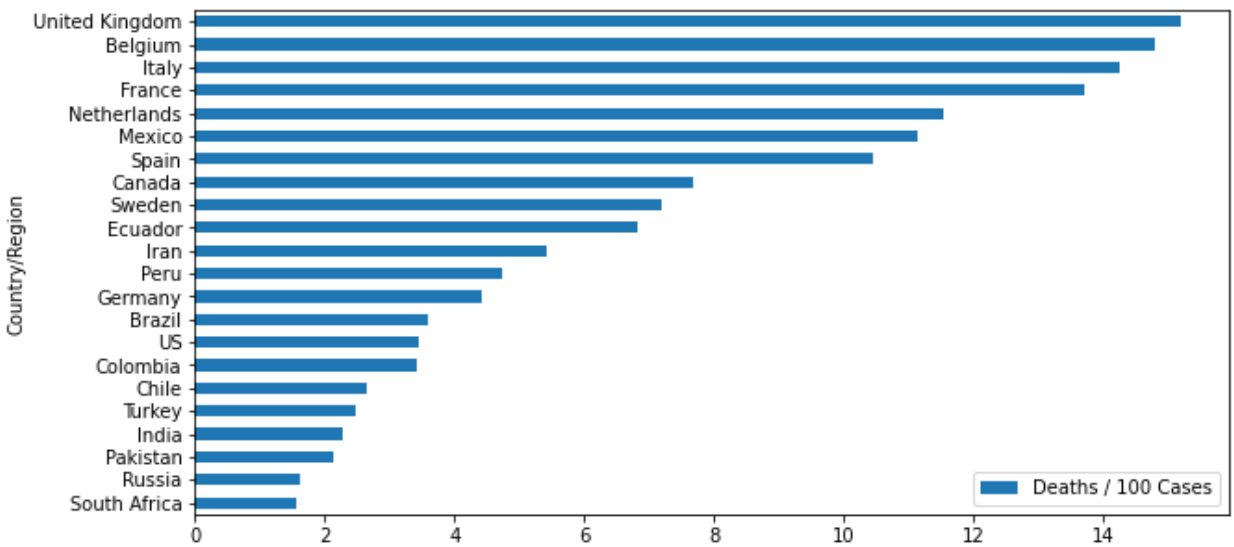


In [318…
```
Image('https://raw.githubusercontent.com/kjmobile/data/main/img/covid_death.PNG
```

Out[318…


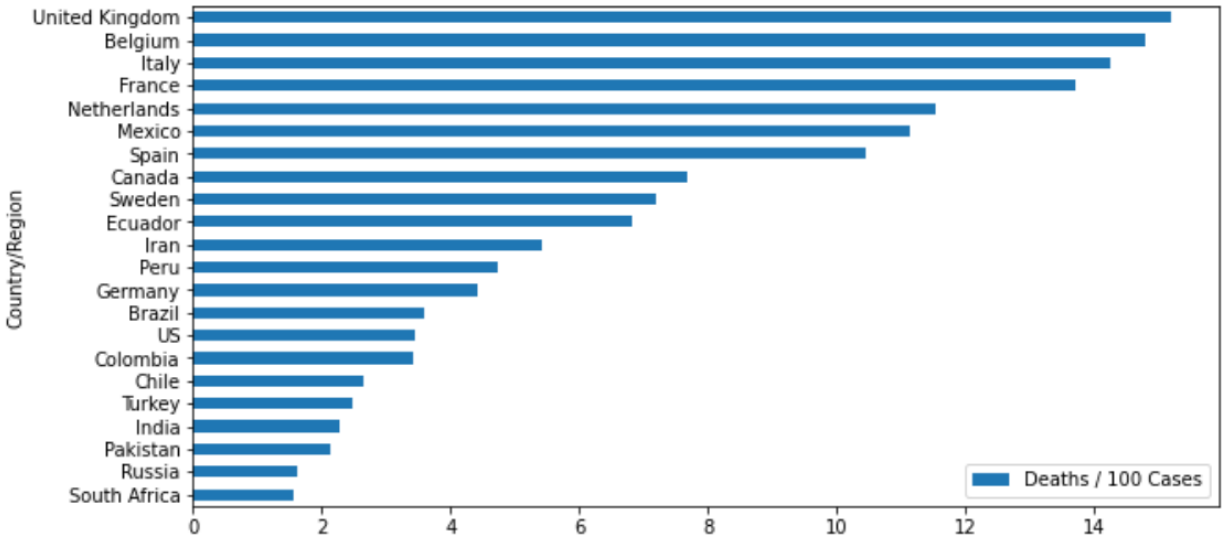
In [319…
```
# 3
covid[covid['Deaths']>5000].sort_values('Deaths / 100 Cases').plot.barh(x='Cou
```

Out[319…
```
<AxesSubplot:ylabel='Country/Region'>
```



In [320…
```
Image('https://raw.githubusercontent.com/kjmobile/data/main/img/covid_death_per
```

Out[320…



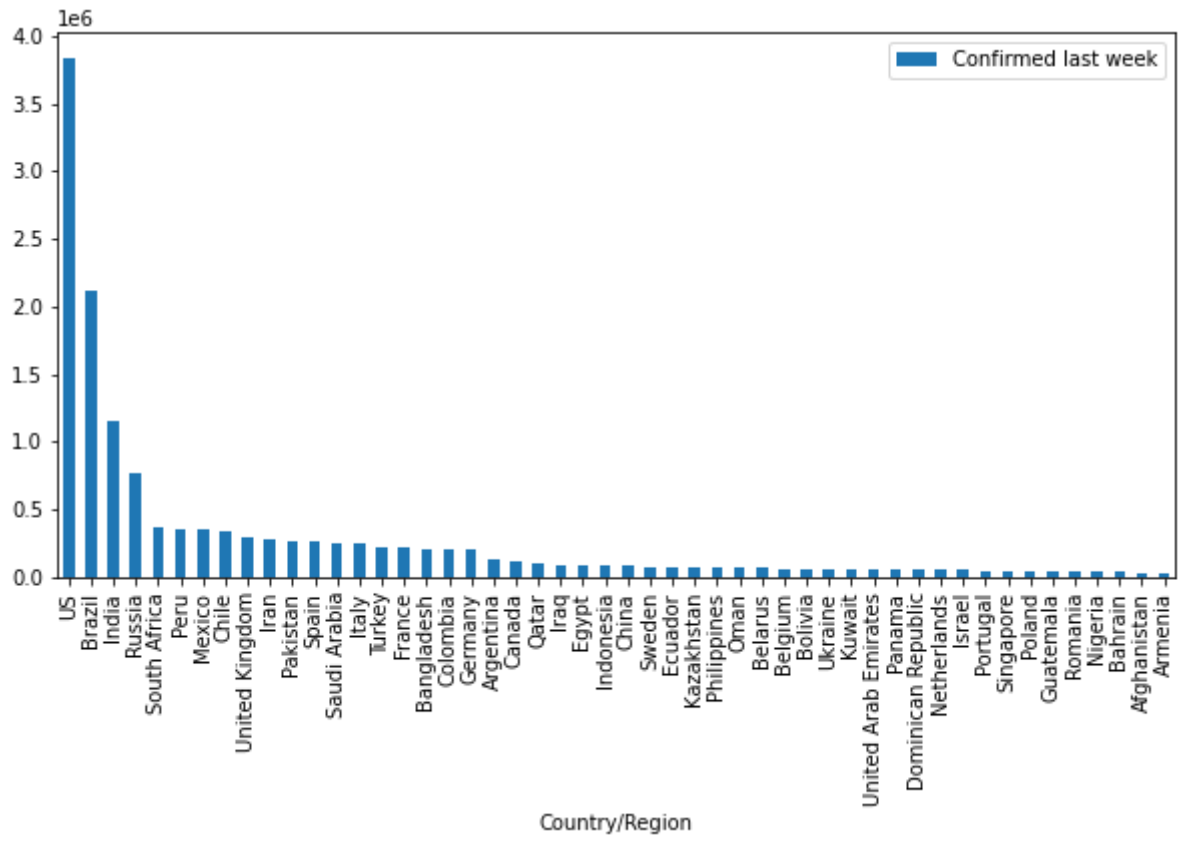
In [321…
```
covid[covid['Country/Region']=='US']
```

Out[321…

	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Re
173	US	4290259	148011	1325804	2816444	56336	1076	27941	3.45	

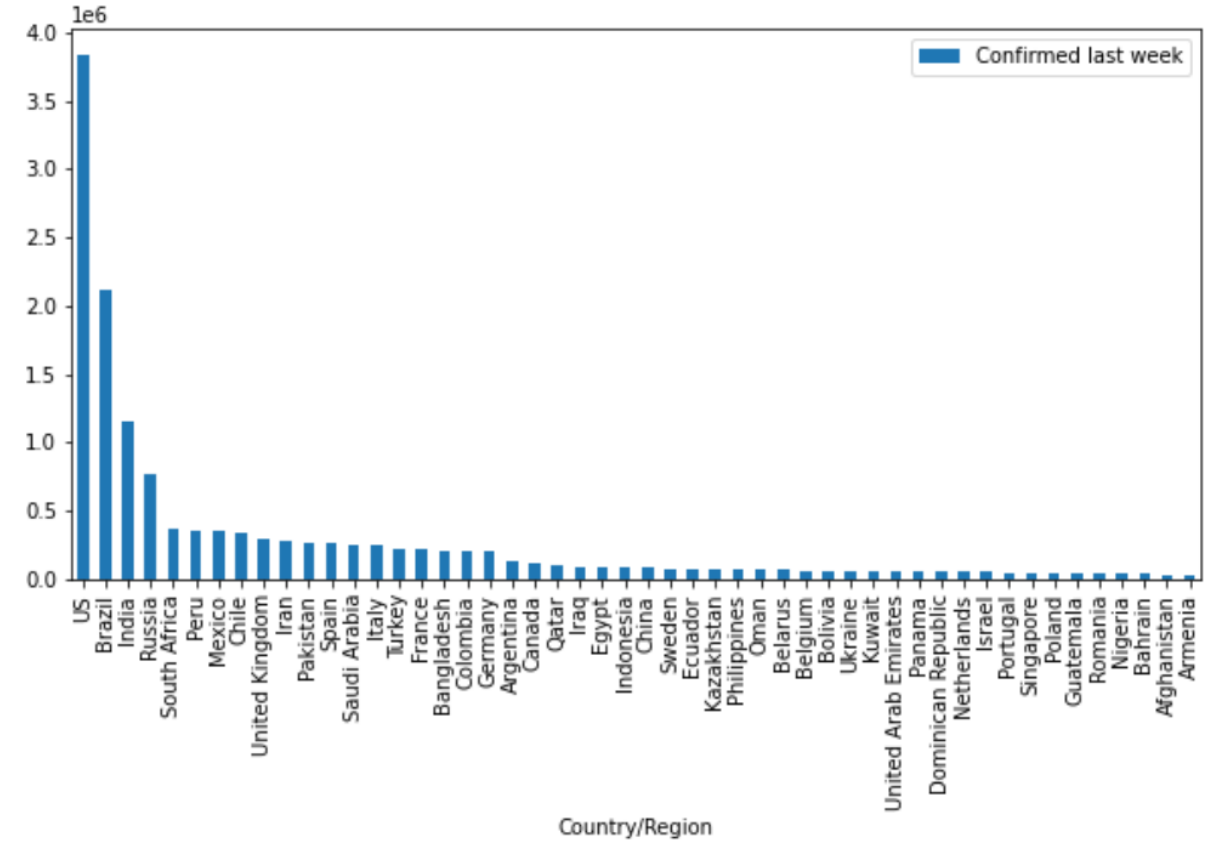
In [322…
```
# 4) Draw a barchart that compares 'Confirmed last week' numbers by country: i
covid.\
sort_values('Confirmed last week', ascending=False).head(50).plot.bar(x='Counti
```

Out[322… <AxesSubplot:xlabel='Country/Region'>



In [323…
```
Image('https://raw.githubusercontent.com/kjmobile/data/main/img/covid_confirmed
```

Out[323…



In [324…
```
covid.columns
```

Out[324…
```
Index(['Country/Region', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'New cases', 'New deaths', 'New recovered', 'Deaths / 100 Cases',
       'Recovered / 100 Cases', 'Deaths / 100 Recovered',
       'Confirmed last week', '1 week change', '1 week % increase',
       'WHO Region'],
      dtype='object')
```

In [325…
```
covid.head(3)
```

Out[325…

	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recover / 1 Ca
0	Afghanistan	36263	1269	25198	9796	106	10	18	3.50	69
1	Albania	4880	144	2745	1991	117	6	63	2.95	56
2	Algeria	27973	1163	18837	7973	616	8	749	4.16	67

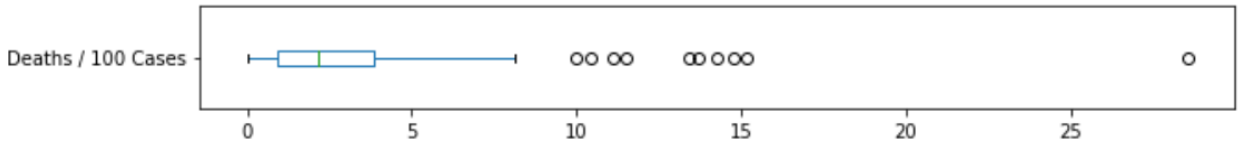
In [326…
```
# 5) Draw a single variable boxplot showing the distribution of 'Deaths / 100 (

covid[['Deaths / 100 Cases']].plot.box(figsize=[10,1], vert=False)
```

Out[326…
```
<AxesSubplot:>
```

In [327…
```
Image('https://raw.githubusercontent.com/kjmobile/data/main/img/covid_boxplot.
```

Out[327…

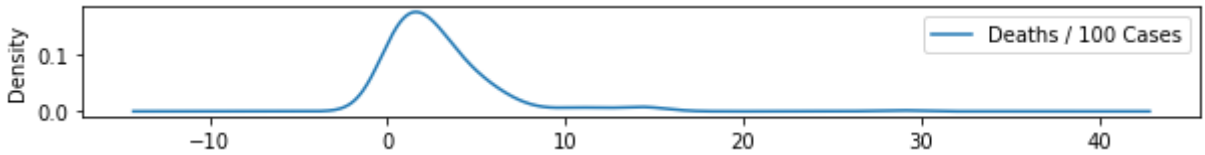


In [328…

```
# 6) Draw of kernel density estimation chart showing the distribution of 'Deati

covid[['Deaths / 100 Cases']].plot.kde(figsize=[10,1])
```

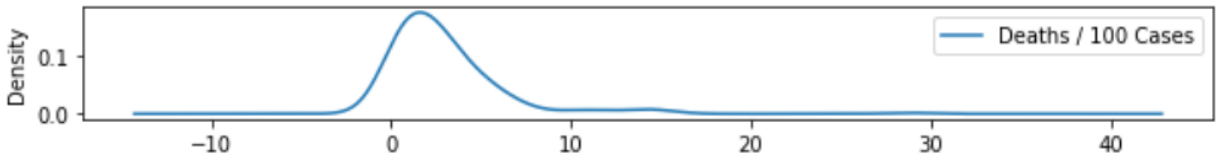
Out[328… <AxesSubplot:ylabel='Density'>



In [329…

```
Image('https://raw.githubusercontent.com/kjmobile/data/main/img/covid_kdeplot.
```

Out[329…



In [ ]:

In [ ]: