Title

Omar Sherif Fouad (Data Science Assignment 2)

Abstract

In this document, the assignment is devided into two parts; (part 1: matches results data)(part 2: covid 19 data). Part 1 is depending on exercise 1, so I have the steps for the first exercise (codes) then the discussion and conclusion section in which I will be analyzing the data. Part 2 doesn't defer from part 1; it is depending on exercise 2, so I have the steps for the second exercise (codes) then comes the discussion and conclusion section in which I will also be have my analyzations.

Steps

Exercise 1

In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv("results.csv")
```

In [3]:
```
df.head(4)
```

Out[3]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
0	1872-11-30	Scotland	England	0	0	Friendly	Glasgow	Scotland	False
1	1873-03-08	England	Scotland	4	2	Friendly	London	England	False
2	1874-03-07	Scotland	England	2	1	Friendly	Glasgow	Scotland	False
3	1875-03-06	England	Scotland	2	2	Friendly	London	England	False

Adding a column win/lose/tie column: We first create a list of conditions

In [4]:
```
x=df['home_score']-df['away_score']
conditions = [
    (x<0),
    (x>0),
    (x==0)
    ]
```

In [5]:
```
values= ['win','lose','draw']
```

In [6]:
```
import numpy as np
```

In [7]:
```
df['result'] = np.select(conditions, values)
```

In [8]:
```
df.head(3)
```

Out[8]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral	result
0	1872-11-30	Scotland	England	0	0	Friendly	Glasgow	Scotland	False	draw
1	1873-03-08	England	Scotland	4	2	Friendly	London	England	False	lose
2	1874-03-07	Scotland	England	2	1	Friendly	Glasgow	Scotland	False	lose

In [9]:
```
df['result'].value_counts()
```

Out[9]:
```
lose    21009
win     12224
draw     9955
Name: result, dtype: int64
```

In [10]:
```
df['result'].value_counts(normalize=True)
```

Out[10]:
```
lose    0.486455
win     0.283042
draw    0.230504
Name: result, dtype: float64
```

In [11]:
```
x=df['result'].value_counts()
```

In [12]:
```
x=np.array(x)
```

In [13]:
```
x
```

Out[13]:
```
array([21009, 12224,  9955])
```

In [14]:
```
N=x.sum()
```

In [15]:
```
N
```

Out[15]:
```
43188
```

In [16]:
```
import statsmodels.api as sm
from statsmodels.stats.proportion import proportion_confint
```

In [17]:
```
CI_win=proportion_confint(count=x[1],nobs=N,alpha=(1-.95))
```

In [18]:
```
CI_win
```

Out[18]:
```
(0.27879305599044235, 0.28729011526083115)
```

In [19]:
```
CI_lose=proportion_confint(count=x[0],nobs=N,alpha=(1-.95))
```

In [20]:
```
CI_lose
```

Out[20]:
```
(0.481740705905987, 0.49116843552218753)
```

In [21]:
```
CI_draw=proportion_confint(count=x[2],nobs=N,alpha=(1-.95))
```

In [22]:
```
CI_draw
```

Out[22]:
```
(0.2265318471530234, 0.23447584016752862)
```

In [23]:
```
df['country'].unique()
```

Out[23]:
```
array(['Scotland', 'England', 'Wales', 'Ireland', 'United States',
       'Uruguay', 'Austria', 'Hungary', 'Argentina', 'Belgium', 'France',
       'Netherlands', 'Bohemia', 'Switzerland', 'Sweden', 'Germany',
       'Italy', 'Chile', 'Norway', 'Finland', 'Luxembourg', 'Russia',
       'Denmark', 'Spain', 'Japan', 'Brazil', 'Paraguay', 'China PR',
       'Canada', 'Estonia', 'Guatemala', 'Czechoslovakia', 'Poland',
       'Yugoslavia', 'New Zealand', 'Romania', 'Latvia', 'Portugal',
       'Northern Ireland', 'Australia', 'Lithuania', 'Turkey', 'Mexico',
       'Aruba', 'Soviet Union', 'Haiti', 'Philippines', 'Bulgaria',
       'Jamaica', 'Kenya', 'Czech Republic', 'Peru', 'Honduras',
       'British Guyana', 'Uganda', 'El Salvador', 'Barbados',
       'Irish Free State', 'Trinidad and Tobago', 'Greece', 'Curaçao',
       'Egypt', 'Dominica', 'Guadeloupe', 'Palestine',
       'Netherlands Guyana', 'French Guiana', 'Cuba', 'Colombia',
       'Saint Kitts and Nevis', 'Éire', 'Panama', 'Bohemia and Moravia',
       'Slovakia', 'Manchuria', 'Croatia', 'Costa Rica', 'Afghanistan',
       'Martinique', 'Southern Rhodesia', 'Iceland', 'Albania',
       'Madagascar', 'Northern Rhodesia', 'Tanganyika', 'Iran', 'Ecuador',
       'French Somaliland', 'Belgian Congo', 'Mauritius', 'Hong Kong',
       'Vietnam', 'Macau', 'Republic of Ireland', 'Ethiopia', 'Suriname',
       'Puerto Rico', 'Réunion', 'Israel', 'Sierra Leone', 'Zanzibar',
       'Bolivia', 'Gold Coast', 'South Africa', 'Netherlands Antilles',
       'New Caledonia', 'Fiji', 'Nigeria', 'Venezuela', 'Ceylon',
       'French Polynesia', 'Gambia', 'Singapore', 'Portuguese Guinea',
       'German DR', 'New Hebrides', 'Burma', 'Saarland', 'Cambodia',
       'India', 'Lebanon', 'Pakistan', 'Malaya', 'South Korea',
       'Vietnam Republic', 'Togo', 'Indonesia', 'Sudan', 'Malta', 'Syria',
       'Tunisia', 'Nyasaland', 'Ghana', 'Morocco', 'United Arab Republic',
       'North Korea', 'Dahomey', 'Guinea-Bissau', 'Mali Federation',
       'Mali', 'Vietnam DR', 'Cyprus', 'Iraq', 'Saint Lucia', 'Senegal',
       'Libya', 'Gabon', 'Thailand', 'Congo', 'Tanzania', 'Grenada',
       'Guinea', 'Central African Republic', 'Cameroon', 'Algeria',
       'Kuwait', 'Ivory Coast', 'Lautoka', 'Liberia', 'Malaysia',
       'Jordan', 'Zambia', 'Saint Vincent and the Grenadines', 'Bermuda',
       'Niger', 'Malawi', 'DR Congo', 'Upper Volta', 'Taiwan', 'Guyana',
       'Mauritania', 'Rhodesia', 'Saudi Arabia', 'Eswatini', 'Mozambique',
       'Papua New Guinea', 'Bahrain', 'Lesotho', 'Somalia', 'Zaïre',
       'Sri Lanka', 'Antigua and Barbuda', 'Faroe Islands', 'Qatar',
       'Yemen DPR', 'Burundi', 'Guam', 'Chad', 'Angola',
       'Dominican Republic', 'Seychelles', 'São Tomé and Príncipe',
       'Botswana', 'Benin', 'Rwanda', 'Bangladesh',
       'United Arab Emirates', 'Zimbabwe', 'Oman', 'Equatorial Guinea',
       'Cape Verde', 'Liechtenstein', 'Nepal', 'Greenland', 'Vanuatu',
       'Western Samoa', 'Belize', 'Brunei', 'Djibouti', 'Burkina Faso',
       'Yemen AR', 'Anguilla', 'Nicaragua', 'Cayman Islands', 'Monaco',
       'Solomon Islands', 'Sint Maarten', 'Namibia', 'Saint Martin',
       'San Marino', 'Slovenia', 'Moldova', 'Ukraine', 'Kazakhstan',
       'Tajikistan', 'Uzbekistan', 'Turkmenistan', 'Georgia',
       'Kyrgyzstan', 'Armenia', 'Belarus', 'Guernsey', 'Azerbaijan',
       'North Macedonia', 'Jersey', 'Montserrat', 'Gibraltar', 'Myanmar',
       'Bosnia and Herzegovina', 'Tonga', 'Andorra', 'Yemen',
       'United States Virgin Islands', 'Palau', 'Cook Islands',
       'British Virgin Islands', 'Eritrea', 'Bahamas', 'Micronesia',
       'Maldives', 'Laos', 'Isle of Man', 'Samoa', 'Bhutan',
       'Serbia and Montenegro', 'Mayotte', 'Mongolia', 'Northern Cyprus',
       'Serbia', 'Montenegro', 'Northern Mariana Islands', 'Comoros',
       'Turks and Caicos Islands', 'South Sudan', 'Kosovo', 'East Timor',
       'Tahiti'], dtype=object)
```

In [24]:
```
dfegy=df[df['country']=='Egypt']
```

In [25]:
```
dfegy.head()
```

Out[25]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral	result
1463	2/19/1932	Egypt	Hungary	0	0	Friendly	Cairo	Egypt	False	draw
1661	3/16/1934	Egypt	Israel	7	1	FIFA World Cup qualification	Cairo	Egypt	False	lose
1895	6/19/1936	Egypt	Greece	3	1	Friendly	Cairo	Egypt	False	lose
2927	12/24/1948	Egypt	Norway	1	1	Friendly	Cairo	Egypt	False	draw
3080	2/17/1950	Egypt	Greece	2	0	Friendly	Cairo	Egypt	False	lose

Create a variable type of the match `friendly/No friendly` and `home/away` match

In [26]:
```
conditions = [
    (dfegy['tournament']=='Friendly'),
    (dfegy['tournament']!='Friendly')
    ]
```

In [27]:
```
values=['Friendly','Official']
```

In [28]:
```
dfegy['typematch'] = np.select(conditions, values)
```

```
/var/folders/yv/41xfsxw90v7gdm2ntzqf4hz40000gn/T/ipykernel_71081/2597929672.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dfegy['typematch'] = np.select(conditions, values)
```

In [29]:
```
dfegy['typematch'].value_counts()
```

Out[29]:
```
Official    226
Friendly    181
Name: typematch, dtype: int64
```

Comparing the probability of win/lose/draw by type of match

In [30]:
```
x=pd.crosstab(dfegy['typematch'],dfegy['result'],margins=True)
x
```

Out[30]:

result	draw	lose	win	All
typematch				
Friendly	37	99	45	181
Official	40	139	47	226
All	77	238	92	407

In [31]:
```
x=np.array(x)
x
```

Out[31]:
```
array([[ 37,  99,  45, 181],
       [ 40, 139,  47, 226],
       [ 77, 238,  92, 407]])
```

In [32]:
```
CI_egywin_friendly=proportion_confint(count=x[0,2],nobs=x[0,3],alpha=(1-.95))
CI_egywin_friendly
```

Out[32]:
```
(0.1856528084886489, 0.3115847605721246)
```

In [33]:
```
CI_egywin_official=proportion_confint(count=x[1,2],nobs=x[1,3],alpha=(1-.95))
CI_egywin_official
```

Out[33]:
```
(0.1550517855722795, 0.2608774179675435)
```

Plotting the Confidence intervals

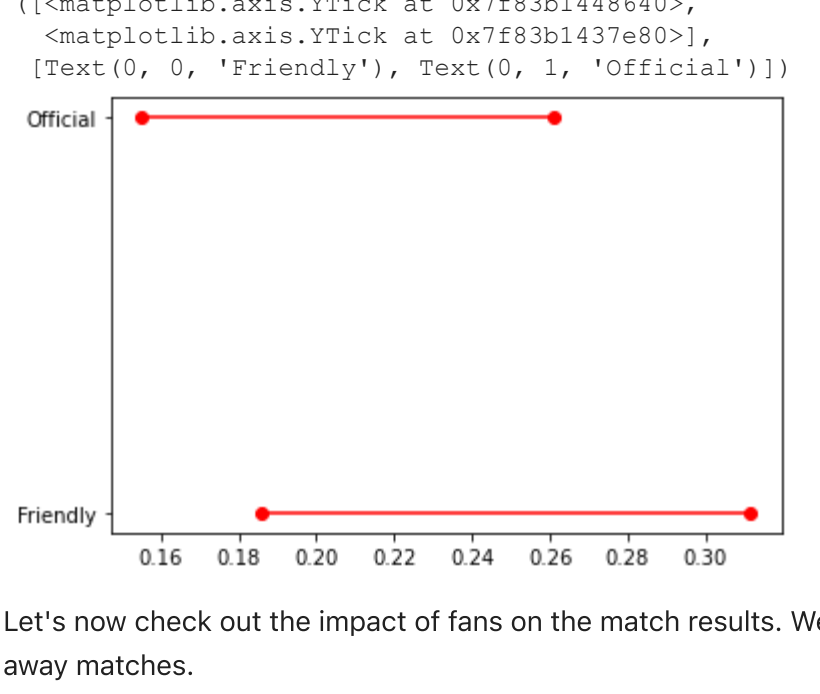
In [34]:
```
ci_egywin = {}
ci_egywin['Typematch'] = ['Friendly','Official']
ci_egywin['lb'] = [CI_egywin_friendly[0],CI_egywin_official[0]]
ci_egywin['ub'] = [CI_egywin_friendly[1],CI_egywin_official[1]]
df_ci= pd.DataFrame(ci_egywin)
df_ci
```

Out[34]:

	Typematch	lb	ub
0	Friendly	0.185653	0.311585
1	Official	0.155052	0.260877

In [35]:
```
import matplotlib.pyplot as plt
for lb,ub,y in zip(df_ci['lb'],df_ci['ub'],range(len(df_ci))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(df_ci)),list(df_ci['Typematch']))
```

Out[35]:
```
([<matplotlib.axis.YTick at 0x7f83b1448640>,
  <matplotlib.axis.YTick at 0x7f83b1437e80>],
 [Text(0, 0, 'Friendly'), Text(0, 1, 'Official')])
```



Let's now check out the impact of fans on the match results. We will compare the probability of winning between home matches and away matches.

In [36]:
```
dfegy['home']=(dfegy['home_team']=='Egypt')
```

```
/var/folders/yv/41xfsxw90v7gdm2ntzqf4hz40000gn/T/ipykernel_71081/2175071764.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dfegy['home']=(dfegy['home_team']=='Egypt')
```

In [37]:
```
dfegy['home'].value_counts()
```

Out[37]:
```
True     258
False    149
Name: home, dtype: int64
```

In [38]:
```
x=pd.crosstab(dfegy['home'],dfegy['result'],margins=True)
x
```

Out[38]:

result	draw	lose	win	All
home				
False	33	67	49	149
True	44	171	43	258
All	77	238	92	407

In [39]:
```
dfegy.head(4)
```

Out[39]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral	result	typematch	home
1463	2/19/1932	Egypt	Hungary	0	0	Friendly	Cairo	Egypt	False	draw	Friendly	True
1661	3/16/1934	Egypt	Israel	7	1	FIFA World Cup qualification	Cairo	Egypt	False	lose	Official	True
1895	6/19/1936	Egypt	Greece	3	1	Friendly	Cairo	Egypt	False	lose	Friendly	True
2927	12/24/1948	Egypt	Norway	1	1	Friendly	Cairo	Egypt	False	draw	Friendly	True

In [40]:
```
x=np.array(x)
x
```

Out[40]:
```
array([[ 33,  67,  49, 149],
       [ 44, 171,  43, 258],
       [ 77, 238,  92, 407]])
```

In [41]:
```
CI_egywin_home=proportion_confint(count=x[1,2],nobs=x[1,3],alpha=(1-.95))
CI_egywin_home
```

Out[41]:
```
(0.12119174183927744, 0.21214159149405587)
```

In [42]:
```
CI_egywin_away=proportion_confint(count=x[0,2],nobs=x[0,3],alpha=(1-.95))
CI_egywin_away
```

Out[42]:
```
(0.25342513144684364, 0.40429298935852553)
```

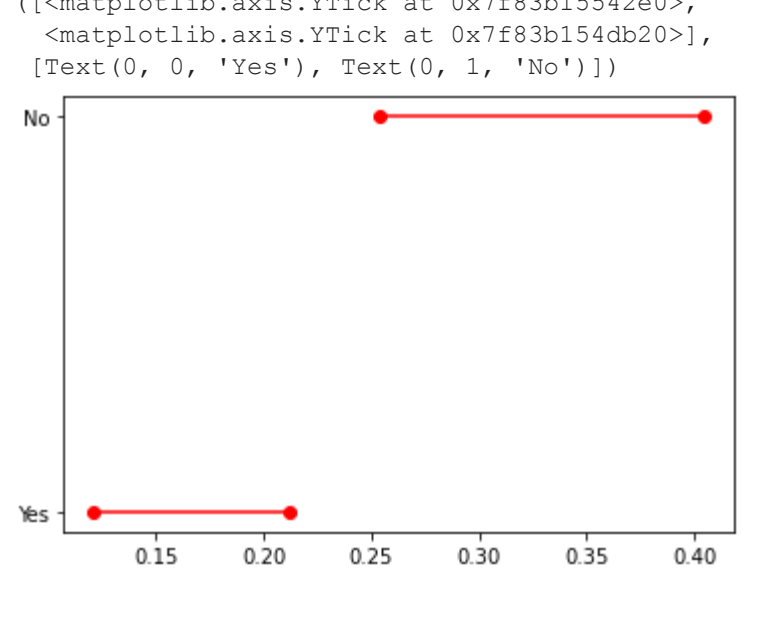
In [43]:
```
ci_egywin = {}
ci_egywin['home'] = ['Yes','No']
ci_egywin['lb'] = [CI_egywin_home[0],CI_egywin_away[0]]
ci_egywin['ub'] = [CI_egywin_home[1],CI_egywin_away[1]]
df_ci= pd.DataFrame(ci_egywin)
df_ci
```

Out[43]:

	home	lb	ub
0	Yes	0.121192	0.212142
1	No	0.253425	0.404293

In [44]:
```
for lb,ub,y in zip(df_ci['lb'],df_ci['ub'],range(len(df_ci))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(df_ci)),list(df_ci['home']))
```

Out[44]:
```
([<matplotlib.axis.YTick at 0x7f83b15542e0>,
  <matplotlib.axis.YTick at 0x7f83b154db20>],
 [Text(0, 0, 'Yes'), Text(0, 1, 'No')])
```



Assignment 2 part 1 (Discussion & Conclusion)

This part will be focussing on:-
winning & losing possibility
difference and chances in home and away matchses
presenting confidence interval and graphs

win and lose possibility

In [45]:
```
x=pd.crosstab(dfegy['typematch'],dfegy['result'],margins=True)
x
```

Out[45]:

result	draw	lose	win	All
typematch				
Friendly	37	99	45	181
Official	40	139	47	226
All	77	238	92	407

This table shows the stats of egypt in all games.
Differing them by the type of the competition.
20% is the winning probability in official games.
25^% is the winning probability in the friendly matches.
Egypt plays better in the friendly games.

home and away stats

In [46]:
```
x=pd.crosstab(dfegy['home'],dfegy['result'],margins=True)
x
```

Out[46]:

result	draw	lose	win	All
home				
False	33	67	49	149
True	44	171	43	258
All	77	238	92	407

This table shows the stats of egypt in all games.
Differing them by home and away games (true: home/false: away).
obviously, egypt's probability of winning in the away matches is much higher.

confidence interval and graphs

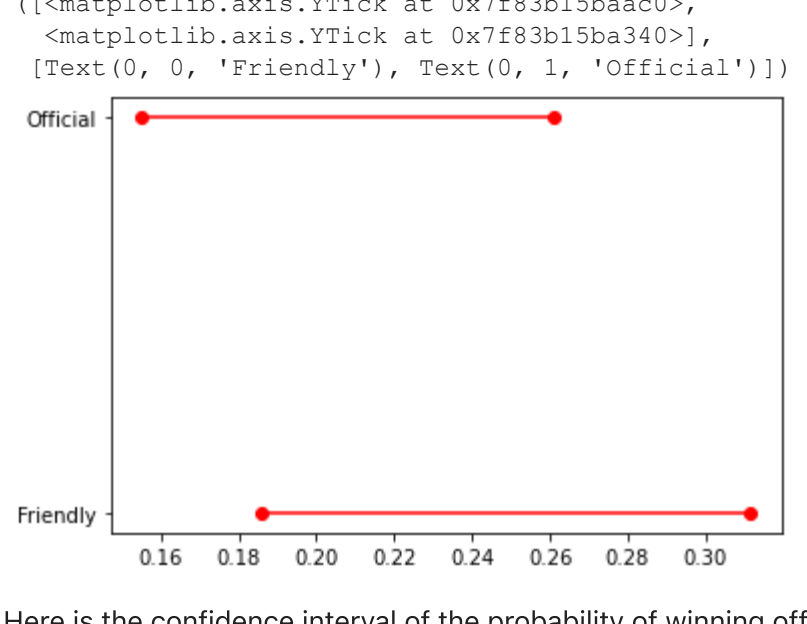
In [47]:
```
ci_egywin = {}
ci_egywin['Typematch'] = ['Friendly','Official']
ci_egywin['lb'] = [CI_egywin_friendly[0],CI_egywin_official[0]]
ci_egywin['ub'] = [CI_egywin_friendly[1],CI_egywin_official[1]]
df_ci= pd.DataFrame(ci_egywin)
df_ci
```

Out[47]:

	Typematch	lb	ub
0	Friendly	0.185653	0.311585
1	Official	0.155052	0.260877

In [48]:
```
import matplotlib.pyplot as plt
for lb,ub,y in zip(df_ci['lb'],df_ci['ub'],range(len(df_ci))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(df_ci)),list(df_ci['Typematch']))
```

Out[48]:
```
([<matplotlib.axis.YTick at 0x7f83b15baac0>,
  <matplotlib.axis.YTick at 0x7f83b15ba340>],
 [Text(0, 0, 'Friendly'), Text(0, 1, 'Official')])
```



Here is the confidence interval of the probability of winning official and friendly matches and under it a graph representing it.
As shown if we have had more matches still egypt has the advantage in the friendly matches.

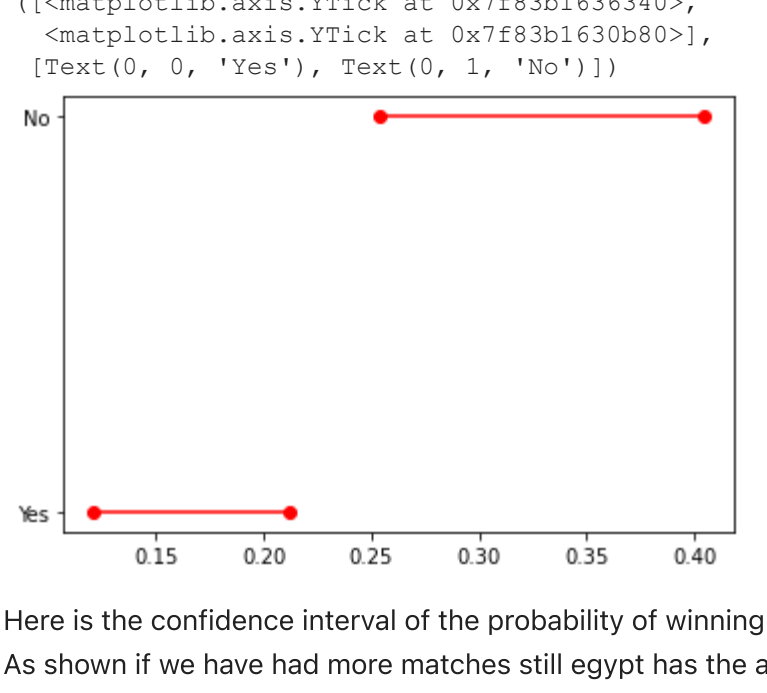
In [49]:
```
ci_egywin = {}
ci_egywin['home'] = ['Yes','No']
ci_egywin['lb'] = [CI_egywin_home[0],CI_egywin_away[0]]
ci_egywin['ub'] = [CI_egywin_home[1],CI_egywin_away[1]]
df_ci= pd.DataFrame(ci_egywin)
df_ci
```

Out[49]:

	home	lb	ub
0	Yes	0.121192	0.212142
1	No	0.253425	0.404293

In [50]:
```
for lb,ub,y in zip(df_ci['lb'],df_ci['ub'],range(len(df_ci))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(df_ci)),list(df_ci['home']))
```

Out[50]:
```
([<matplotlib.axis.YTick at 0x7f83b1636340>,
  <matplotlib.axis.YTick at 0x7f83b1630b80>],
 [Text(0, 0, 'Yes'), Text(0, 1, 'No')])
```



Here is the confidence interval of the probability of winning home and away matches and under it a graph representing it.
As shown if we have had more matches still egypt has the advantage in the away matches.

Exercise 2

In [51]:
```
import pandas as pd
```

In [52]:
```
df=pd.read_csv('covid_data.csv',encoding='latin-1')
```

In [53]:
```
df.head()
```

Out[53]:

	date	iso3c	country	income	region	continent	dcases	ddeaths	population	weekdays	month
0	2020-02-24	AFG	Afghanistan	Low income	South Asia	Asia	5	0	38041754	Mon	Feb
1	2020-02-25	AFG	Afghanistan	Low income	South Asia	Asia	0	0	38041754	Tue	Feb
2	2020-02-26	AFG	Afghanistan	Low income	South Asia	Asia	0	0	38041754	Wed	Feb
3	2020-02-27	AFG	Afghanistan	Low income	South Asia	Asia	0	0	38041754	Thu	Feb
4	2020-02-28	AFG	Afghanistan	Low income	South Asia	Asia	0	0	38041754	Fri	Feb

In [54]:
```
df['country'].unique()
```

Out[54]:
```
array(['Afghanistan', 'Angola', 'Albania', 'Andorra',
       'United Arab Emirates', 'Argentina', 'Armenia',
       'Antigua & Barbuda', 'Australia', 'Austria', 'Azerbaijan',
       'Burundi', 'Belgium', 'Benin', 'Burkina Faso', 'Bangladesh',
       'Bulgaria', 'Bahrain', 'Bahamas', 'Bosnia & Herzegovina',
       'Belarus', 'Belize', 'Bolivia', 'Brazil', 'Barbados', 'Brunei',
       'Bhutan', 'Botswana', 'Central African Republic', 'Canada',
       'Switzerland', 'Chile', 'China', 'Côte d\x92Ivoire', 'Cameroon',
       'Congo - Kinshasa', 'Congo - Brazzaville', 'Colombia', 'Comoros',
       'Cape Verde', 'Costa Rica', 'Cuba', 'Cyprus', 'Czechia', 'Germany',
       'Djibouti', 'Dominica', 'Denmark', 'Dominican Republic', 'Algeria',
       'Ecuador', 'Egypt', 'Spain', 'Estonia', 'Ethiopia', 'Finland',
       'Fiji', 'France', 'Gabon', 'United Kingdom', 'Georgia', 'Ghana',
       'Guinea', 'Gambia', 'Guinea-Bissau', 'Equatorial Guinea', 'Greece',
       'Grenada', 'Guatemala', 'Guyana', 'Honduras', 'Croatia', 'Haiti',
       'Hungary', 'Indonesia', 'India', 'Ireland', 'Iran', 'Iraq',
       'Iceland', 'Israel', 'Italy', 'Jamaica', 'Jordan', 'Japan',
       'Kazakhstan', 'Kenya', 'Kyrgyzstan', 'Cambodia', 'Kiribati',
       'St. Kitts & Nevis', 'South Korea', 'Kuwait', 'Laos', 'Lebanon',
       'Liberia', 'Libya', 'St. Lucia', 'Liechtenstein', 'Sri Lanka',
       'Lesotho', 'Lithuania', 'Luxembourg', 'Latvia', 'Morocco',
       'Monaco', 'Moldova', 'Madagascar', 'Maldives', 'Mexico',
       'Marshall Islands', 'North Macedonia', 'Mali', 'Malta',
       'Myanmar (Burma)', 'Montenegro', 'Mongolia', 'Mozambique',
       'Mauritania', 'Mauritius', 'Malawi', 'Malaysia', 'Namibia',
       'Niger', 'Nigeria', 'Nicaragua', 'Netherlands', 'Norway', 'Nepal',
       'New Zealand', 'Oman', 'Pakistan', 'Panama', 'Peru', 'Philippines',
       'Palau', 'Papua New Guinea', 'Poland', 'Portugal', 'Paraguay',
       'Palestinian Territories', 'Qatar', 'Romania', 'Russia', 'Rwanda',
       'Saudi Arabia', 'Sudan', 'Senegal', 'Singapore', 'Solomon Islands',
       'Sierra Leone', 'El Salvador', 'San Marino', 'Somalia', 'Serbia',
       'South Sudan', 'São Tomé & Príncipe', 'Suriname', 'Slovakia',
       'Slovenia', 'Sweden', 'Eswatini', 'Seychelles', 'Syria', 'Chad',
       'Togo', 'Thailand', 'Tajikistan', 'Timor-Leste',
       'Trinidad & Tobago', 'Tunisia', 'Turkey', 'Tanzania', 'Uganda',
       'Ukraine', 'Uruguay', 'United States', 'Uzbekistan',
       'St. Vincent & Grenadines', 'Venezuela', 'Vietnam', 'Vanuatu',
       'Samoa', 'Yemen', 'South Africa', 'Zambia', 'Zimbabwe'],
      dtype=object)
```

In [55]:
```
from pandas.api.types import CategoricalDtype
cats=['Fri', 'Sat', 'Sun','Mon','Tue','Wed','Thu']
cat_type = CategoricalDtype(categories=cats, ordered=True)
df['weekdays'] = df['weekdays'].astype(cat_type)
```

In [56]:
```
from pandas.api.types import CategoricalDtype
cats=['Jan', 'Feb', 'Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
cat_type = CategoricalDtype(categories=cats, ordered=True)
df['month'] = df['month'].astype(cat_type)
```

In [57]:
```
import numpy as np
```

In [58]:
```
dfegy=df[df['country']=='Egypt']
```

In [59]:
```
stats=dfegy.groupby("weekdays").agg({"dcases": [np.mean, np.std, np.size]})
```

In [60]:
```
stats
```

Out[60]:

	dcases		
	mean	std	size
weekdays			
Fri	567.161616	428.533849	99
Sat	558.806122	421.803605	98
Sun	545.520408	422.358748	98
Mon	561.846939	442.137949	98
Tue	566.153061	419.125460	98
Wed	561.479592	406.337812	98
Thu	567.683673	410.020004	98

In [61]:
```
statsm=dfegy.groupby("month").agg({"dcases": [np.mean, np.std, np.size]})
statsm
```

Out[61]:

	dcases		
	mean	std	size
month			
Jan	899.645161	251.335970	31
Feb	374.409091	288.087868	44
Mar	329.290323	310.292849	62
Apr	504.133333	354.246465	60
May	879.774194	346.475245	62
Jun	1032.633333	454.674216	60
Jul	463.661290	437.595552	62
Aug	145.806452	60.683224	62
Sep	339.033333	225.855584	60
Oct	497.580645	362.321547	62
Nov	598.616667	370.082496	60
Dec	792.709677	300.346870	62

In [62]:
```
import numpy as np
import scipy.stats
```

In [63]:
```
statsm.index
```

Out[63]:
```
CategoricalIndex(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
                  'Sep', 'Oct', 'Nov', 'Dec'],
                 categories=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', ...], ordered=True, dtype='category', name='month')
```

In [64]:
```
ci95_hi = []
ci95_lo = []
```

In [65]:
```
for i in statsm.index:
    m, s, n = statsm.loc[i]
    x=scipy.stats.t.interval(.95, n-1, m,s/np.sqrt(n-1))
    ci95_hi.append(x[1])
    ci95_lo.append(x[0])
```

In [66]:
```
ci95_hi
```

Out[66]:
```
[993.3598571863843,
 463.0084047471138,
 408.7332244407111,
 596.4171621669087,
 968.4807139392825,
 1151.0793353591891,
 575.6969386179874,
 161.3429084204986,
 397.87038622259695,
 590.3442237843955,
 695.0258988600798,
 869.6061543814164]
```

In [67]:
```
ci95_lo
```

Out[67]:
```
[805.9304653942609,
 285.8097770710681,
 249.84742072057924,
 411.8495044997579,
 791.0676731574916,
 914.1873313074777,
 351.6256420271739,
 130.26999480530787,
 280.19628044406977,
 404.8170665381852,
 502.2074344732536,
 715.8132004572933]
```

In [68]:
```
statsm['ci95_hi'] = ci95_hi
statsm['ci95_lo'] = ci95_lo
print(statsm)
```

```
           dcases                          ci95_hi     ci95_lo
             mean         std size
month
Jan     899.645161  251.335970   31   993.359857  805.930465
Feb     374.409091  288.087868   44   463.008405  285.809777
Mar     329.290323  310.292849   62   408.733224  249.847421
Apr     504.133333  354.246465   60   596.417162  411.849504
May     879.774194  346.475245   62   968.480714  791.067673
Jun    1032.633333  454.674216   60  1151.079335  914.187331
Jul     463.661290  437.595552   62   575.696939  351.625642
Aug     145.806452   60.683224   62   161.342908  130.269995
Sep     339.033333  225.855584   60   397.870386  280.196280
Oct     497.580645  362.321547   62   590.344224  404.817067
Nov     598.616667  370.082496   60   695.025899  502.207434
Dec     792.709677  300.346870   62   869.606154  715.813200
```

In [69]:
```
df_ci= pd.DataFrame(statsm)
```

In [70]:
```
df_ci
```

Out[70]:

	dcases			ci95_hi	ci95_lo
	mean	std	size		
month					
Jan	899.645161	251.335970	31	993.359857	805.930465
Feb	374.409091	288.087868	44	463.008405	285.809777
Mar	329.290323	310.292849	62	408.733224	249.847421
Apr	504.133333	354.246465	60	596.417162	411.849504
May	879.774194	346.475245	62	968.480714	791.067673
Jun	1032.633333	454.674216	60	1151.079335	914.187331
Jul	463.661290	437.595552	62	575.696939	351.625642
Aug	145.806452	60.683224	62	161.342908	130.269995
Sep	339.033333	225.855584	60	397.870386	280.196280
Oct	497.580645	362.321547	62	590.344224	404.817067
Nov	598.616667	370.082496	60	695.025899	502.207434
Dec	792.709677	300.346870	62	869.606154	715.813200

In [71]:
```
df_ci.index
```

Out[71]:
```
CategoricalIndex(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
                  'Sep', 'Oct', 'Nov', 'Dec'],
                 categories=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', ...], ordered=True, dtype='category', name='month')
```

In [72]:
```
df_ci['month']=df_ci.index
```

In [73]:
```
df_ci
```

Out[73]:

	dcases			ci95_hi	ci95_lo	month
	mean	std	size			
month						
Jan	899.645161	251.335970	31	993.359857	805.930465	Jan
Feb	374.409091	288.087868	44	463.008405	285.809777	Feb
Mar	329.290323	310.292849	62	408.733224	249.847421	Mar
Apr	504.133333	354.246465	60	596.417162	411.849504	Apr
May	879.774194	346.475245	62	968.480714	791.067673	May
Jun	1032.633333	454.674216	60	1151.079335	914.187331	Jun
Jul	463.661290	437.595552	62	575.696939	351.625642	Jul
Aug	145.806452	60.683224	62	161.342908	130.269995	Aug
Sep	339.033333	225.855584	60	397.870386	280.196280	Sep
Oct	497.580645	362.321547	62	590.344224	404.817067	Oct
Nov	598.616667	370.082496	60	695.025899	502.207434	Nov
Dec	792.709677	300.346870	62	869.606154	715.813200	Dec

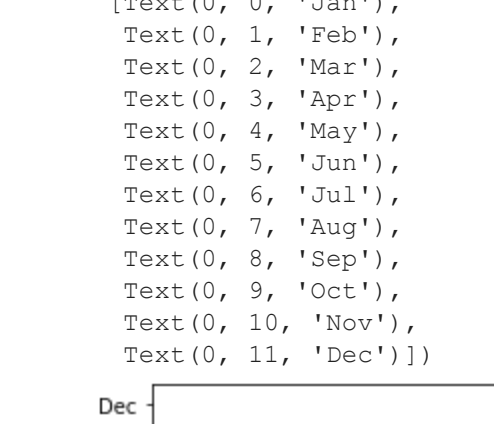
In [74]:
```
df_ci.columns
```

Out[74]:
```
MultiIndex([( 'dcases', 'mean'),
            ( 'dcases',  'std'),
            ( 'dcases', 'size'),
            ('ci95_hi',     ''),
            ('ci95_lo',     ''),
            (  'month',     '')],
           )
```

In [75]:
```
import matplotlib.pyplot as plt
```

In [76]:
```
for lb,ub,y in zip(df_ci['ci95_lo'],df_ci['ci95_hi'],range(len(df_ci))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(df_ci)),list(df_ci['month']))
```

```
Out[76]: ([<matplotlib.axis.YTick at 0x7f83b3c87cd0>,
  <matplotlib.axis.YTick at 0x7f83b3c87550>,
  <matplotlib.axis.YTick at 0x7f83b3c803d0>,
  <matplotlib.axis.YTick at 0x7f83b043a0d0>,
  <matplotlib.axis.YTick at 0x7f83b042fca0>,
  <matplotlib.axis.YTick at 0x7f83b043aac0>,
  <matplotlib.axis.YTick at 0x7f83b10af250>,
  <matplotlib.axis.YTick at 0x7f83b10af9a0>,
  <matplotlib.axis.YTick at 0x7f83b10b4130>,
  <matplotlib.axis.YTick at 0x7f83b10b4880>,
  <matplotlib.axis.YTick at 0x7f83b10b4b80>,
  <matplotlib.axis.YTick at 0x7f83b10af8e0>],
 [Text(0, 0, 'Jan'),
  Text(0, 1, 'Feb'),
  Text(0, 2, 'Mar'),
  Text(0, 3, 'Apr'),
  Text(0, 4, 'May'),
  Text(0, 5, 'Jun'),
  Text(0, 6, 'Jul'),
  Text(0, 7, 'Aug'),
  Text(0, 8, 'Sep'),
  Text(0, 9, 'Oct'),
  Text(0, 10, 'Nov'),
  Text(0, 11, 'Dec')])
```



COVID Cases by Month

In [77]:
```
import pandas as pd
```

In [78]:
```
df=pd.read_csv('covid_data.csv',encoding='latin-1')
```

In [79]:
```
from pandas.api.types import CategoricalDtype
cats=['Jan', 'Feb', 'Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
cat_type = CategoricalDtype(categories=cats, ordered=True)
df['month'] = df['month'].astype(cat_type)
```

In [80]:
```
dfegy=df[df['country']=='Egypt']
```

In [81]:
```
import numpy as np
```

In [82]:
```
stats=dfegy.groupby("month").agg({"dcases": [np.mean, np.std, np.size]})
```

In [83]:
```
stats=pd.DataFrame(stats)
```

In [84]:
```
stats.shape
```

Out[84]:
```
(12, 3)
```

In [85]:
```
stats.columns
```

Out[85]:
```
MultiIndex([('dcases', 'mean'),
            ('dcases',  'std'),
            ('dcases', 'size')],
           )
```

In [86]:
```
stats.columns=['mean','std','size']
```

In [87]:
```
stats.columns
```

Out[87]:
```
Index(['mean', 'std', 'size'], dtype='object')
```

In [88]:
```
def get_ci_lb(x, alpha=0.05):
    sample_s=np.std(x)
    sample_mean=np.mean(x)
    sample_size=len(x)
    margin_of_error = t.ppf(1 - alpha/2,sample_size-1)*sample_s/np.sqrt(sample_size-1)
    return sample_mean - margin_of_error
```

In [89]:
```
x=dfegy['dcases']
```

In [90]:
```
from scipy.stats import norm,t
```

In [91]:
```
get_ci_lb(x)
```

Out[91]:
```
529.7945911276133
```

In [92]:
```
def get_ci_ub(x, alpha=0.05):
    sample_s=np.std(x)
    sample_mean=np.mean(x)
    sample_size=len(x)
    margin_of_error = t.ppf(1 - alpha/2,sample_size-1)*sample_s/np.sqrt(sample_size-1)
    return sample_mean + margin_of_error
```

In [93]:
```
get_ci_ub(x)
```

Out[93]:
```
592.694491841819
```

In [94]:
```
ci_dcases=statsm=dfegy.groupby("month").agg({"dcases": [np.mean, np.std, np.size,get_ci_lb,get_ci_ub]})
statsm
```

Out[94]:

	dcases				
	mean	std	size	get_ci_lb	get_ci_ub
month					
Jan	899.645161	251.335970	31	807.454383	991.835939
Feb	374.409091	288.087868	44	286.822374	461.995808
Mar	329.290323	310.292849	62	250.490694	408.089951
Apr	504.133333	354.246465	60	412.621768	595.644899
May	879.774194	346.475245	62	791.785956	967.762431
Jun	1032.633333	454.674216	60	915.178529	1150.088138
Jul	463.661290	437.595552	62	352.532828	574.789752
Aug	145.806452	60.683224	62	130.395798	161.217105
Sep	339.033333	225.855584	60	280.688649	397.378017
Oct	497.580645	362.321547	62	405.568201	589.593089
Nov	598.616667	370.082496	60	503.014220	694.219113
Dec	792.709677	300.346870	62	716.435854	868.983501

In [95]:
```
statsm.index
```

Out[95]:
```
CategoricalIndex(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
                  'Sep', 'Oct', 'Nov', 'Dec'],
                 categories=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', ...], ordered=True, dtype
='category', name='month')
```

In [96]:
```
statsm.columns=['mean','std','size','lb','ub']
```

In [97]:
```
statsm['month']=statsm.index
```

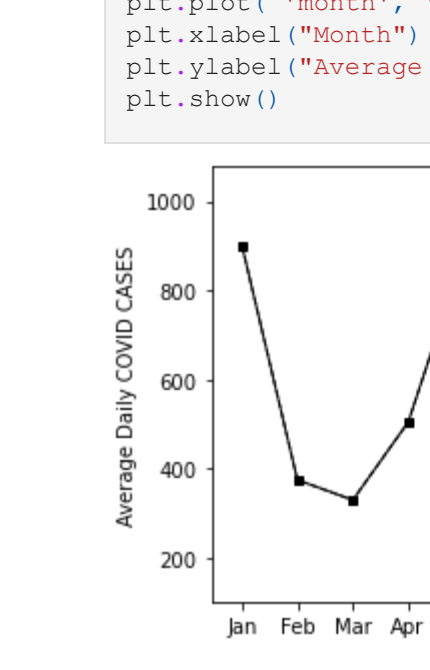
In [98]:
```
statsm
```

Out[98]:

	mean	std	size	lb	ub	month
month						
Jan	899.645161	251.335970	31	807.454383	991.835939	Jan
Feb	374.409091	288.087868	44	286.822374	461.995808	Feb
Mar	329.290323	310.292849	62	250.490694	408.089951	Mar
Apr	504.133333	354.246465	60	412.621768	595.644899	Apr
May	879.774194	346.475245	62	791.785956	967.762431	May
Jun	1032.633333	454.674216	60	915.178529	1150.088138	Jun
Jul	463.661290	437.595552	62	352.532828	574.789752	Jul
Aug	145.806452	60.683224	62	130.395798	161.217105	Aug
Sep	339.033333	225.855584	60	280.688649	397.378017	Sep
Oct	497.580645	362.321547	62	405.568201	589.593089	Oct
Nov	598.616667	370.082496	60	503.014220	694.219113	Nov
Dec	792.709677	300.346870	62	716.435854	868.983501	Dec

In [99]:
```
import matplotlib.pyplot as plt
```

In [100…
```
plt.plot( 'month', 'mean', data=statsm, marker='s', color='black', markersize=4,     linewidth=1, linestyle='--
plt.plot( 'month', 'mean', data=statsm, marker='o', color='black',  markersize=4, linewidth=1,linestyle='-')
plt.xlabel("Month")
plt.ylabel("Average Daily COVID CASES")
plt.show()
```



We will do the same graph now but only on the year 2021. We will first create the column `year`. It works as follows:

1. We first tell to Python that the `date` column is composed of dates object
2. We can then tell to Python to extract the year from the date

In [101…
```
df['date'][0]
```

Out[101…
```
'2020-02-24'
```

In [102…
```
df['date'] = pd. to_datetime(df['date'],format='%Y-%m-%d')
```

In [103…
```
df['date'][0]
```

Out[103…
```
Timestamp('2020-02-24 00:00:00')
```

In [104…
```
df['year'] = pd. DatetimeIndex(df['date']). year
```

In [105…
```
df['year'][0]
```

Out[105…
```
2020
```

I compute now, by country, by year, by month the following statistics on the daily COVID cases: mean, std, size, CI(95%) LB, and UB.

In [106…
```
statsdcases=df.groupby(['country','year','month']).agg({"dcases": [np.mean, np.std, np.size,get_ci_lb,get_ci_ub
```

In [107…
```
statsdcases
```

Out[107…

			dcases				
			mean	std	size	get_ci_lb	get_ci_ub
country	year	month					
Afghanistan	2020	Jan	NaN	NaN	NaN	NaN	NaN
		Feb	0.833333	2.041241	6.0	-1.308818	2.975485
		Mar	5.258065	10.871883	31.0	1.270225	9.245904
		Apr	55.366667	40.385627	30.0	40.286426	70.446908
		May	430.741935	266.692078	31.0	332.918491	528.565379
...	...	...	...	...	...	...	...
Zimbabwe	2021	Aug	513.322581	386.841948	31.0	371.427809	655.217353
		Sep	201.566667	135.119789	30.0	151.112108	252.021225
		Oct	69.580645	58.035492	31.0	48.293055	90.868235
		Nov	54.933333	82.622087	30.0	24.081739	85.784928
		Dec	2536.548387	2572.199964	31.0	1593.057823	3480.038951

4488 rows × 5 columns

Transform the data index to columns

In [108…
```
statsdcases=statsdcases.reset_index()
```

In [109…
```
statsdcases
```

Out[109…

	country	year	month	dcases				
				mean	std	size	get_ci_lb	get_ci_ub
0	Afghanistan	2020	Jan	NaN	NaN	NaN	NaN	NaN
1	Afghanistan	2020	Feb	0.833333	2.041241	6.0	-1.308818	2.975485
2	Afghanistan	2020	Mar	5.258065	10.871883	31.0	1.270225	9.245904
3	Afghanistan	2020	Apr	55.366667	40.385627	30.0	40.286426	70.446908
4	Afghanistan	2020	May	430.741935	266.692078	31.0	332.918491	528.565379
...	...	...	...	...	...	...	...	...
4483	Zimbabwe	2021	Aug	513.322581	386.841948	31.0	371.427809	655.217353
4484	Zimbabwe	2021	Sep	201.566667	135.119789	30.0	151.112108	252.021225
4485	Zimbabwe	2021	Oct	69.580645	58.035492	31.0	48.293055	90.868235
4486	Zimbabwe	2021	Nov	54.933333	82.622087	30.0	24.081739	85.784928
4487	Zimbabwe	2021	Dec	2536.548387	2572.199964	31.0	1593.057823	3480.038951

4488 rows × 8 columns

Filter now the Egypt data on 2021

In [110…
```
statsEgy=statsdcases[(statsdcases['country']=='Egypt') & (statsdcases['year']==2021)]
```

In [111…
```
statsEgy
```

Out[111…

	country	year	month	dcases				
				mean	std	size	get_ci_lb	get_ci_ub
1236	Egypt	2021	Jan	899.645161	251.335970	31.0	807.454383	991.835939
1237	Egypt	2021	Feb	588.321429	40.989659	28.0	572.427298	604.215559
1238	Egypt	2021	Mar	635.709677	36.309038	31.0	622.391415	649.027940
1239	Egypt	2021	Apr	847.366667	92.416405	30.0	812.857814	881.875520
1240	Egypt	2021	May	1132.193548	55.087457	31.0	1111.987306	1152.399791
1241	Egypt	2021	Jun	621.066667	197.770841	30.0	547.217821	694.915513
1242	Egypt	2021	Jul	96.129032	61.659139	31.0	73.512277	118.745787
1243	Egypt	2021	Aug	134.806452	73.635326	31.0	107.796796	161.816107
1244	Egypt	2021	Sep	536.100000	151.276488	30.0	479.612431	592.587569
1245	Egypt	2021	Oct	854.612903	53.664189	31.0	834.928720	874.297087
1246	Egypt	2021	Nov	918.700000	248.217528	30.0	826.014052	1011.385948
1247	Egypt	2021	Dec	870.870968	212.967093	31.0	792.754007	948.987928

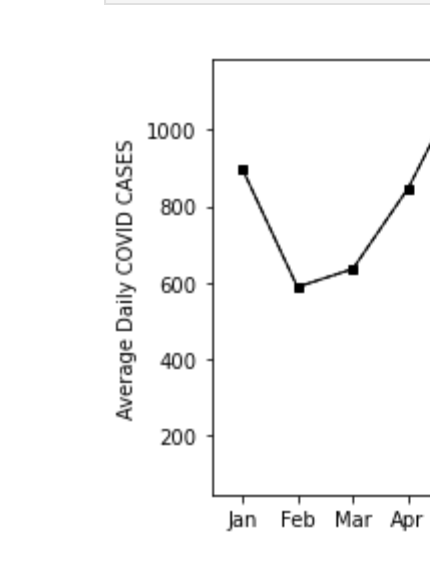
In [112…
```
statsEgy.columns
```

Out[112…
```
MultiIndex([('country',          ''),
            (   'year',          ''),
            (  'month',          ''),
            ( 'dcases',      'mean'),
            ( 'dcases',       'std'),
            ( 'dcases',      'size'),
            ( 'dcases', 'get_ci_lb'),
            ( 'dcases', 'get_ci_ub')],
           )
```

In [113…
```
statsEgy.columns=['country','year','month','mean','std','size','lb','ub']
```

Draw then the average daily COVID cases by month in Egypt.

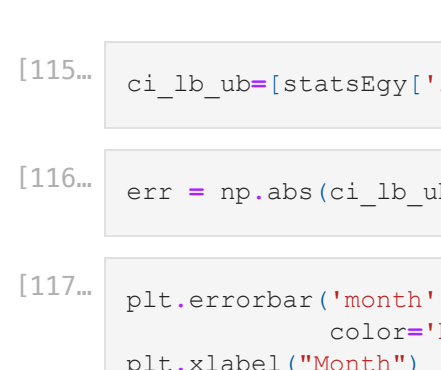
In [114…
```
plt.plot( 'month', 'mean', data=statsEgy, marker='s', color='black', markersize=4,     linewidth=1, linestyle='
plt.plot( 'month', 'mean', data=statsEgy, marker='o', color='black',  markersize=4, linewidth=1,linestyle='-')
plt.xlabel("Month")
plt.ylabel("Average Daily COVID CASES")
plt.title("Egypt")
plt.show()
```



In [115…
```
ci_lb_ub=[statsEgy['lb'],statsEgy['ub']]
```

In [116…
```
err = np.abs(ci_lb_ub - statsEgy['mean'].to_numpy())
```

In [117…
```
plt.errorbar('month', 'mean',  yerr=err, data=statsEgy,marker='s', capsize=2,
             color='black', markersize=4, linewidth=1, linestyle='--')
plt.xlabel("Month")
plt.ylabel("Average Daily COVID CASES")
plt.title("Egypt")
plt.show()
```



Let's compare now between 2020 and 2021

In [118…
```
statsEgy21=statsdcases[(statsdcases['country']=='Egypt')
                      & (statsdcases['year']==2021)]
statsEgy21
```

Out[118…

	country	year	month	dcases				
				mean	std	size	get_ci_lb	get_ci_ub
1236	Egypt	2021	Jan	899.645161	251.335970	31.0	807.454383	991.835939
1237	Egypt	2021	Feb	588.321429	40.989659	28.0	572.427298	604.215559
1238	Egypt	2021	Mar	635.709677	36.309038	31.0	622.391415	649.027940
1239	Egypt	2021	Apr	847.366667	92.416405	30.0	812.857814	881.875520
1240	Egypt	2021	May	1132.193548	55.087457	31.0	1111.987306	1152.399791
1241	Egypt	2021	Jun	621.066667	197.770841	30.0	547.217821	694.915513
1242	Egypt	2021	Jul	96.129032	61.659139	31.0	73.512277	118.745787
1243	Egypt	2021	Aug	134.806452	73.635326	31.0	107.796796	161.816107
1244	Egypt	2021	Sep	536.100000	151.276488	30.0	479.612431	592.587569
1245	Egypt	2021	Oct	854.612903	53.664189	31.0	834.928720	874.297087
1246	Egypt	2021	Nov	918.700000	248.217528	30.0	826.014052	1011.385948
1247	Egypt	2021	Dec	870.870968	212.967093	31.0	792.754007	948.987928

In [119…
```
statsEgy20=statsdcases[(statsdcases['country']=='Egypt')
                      & (statsdcases['year']==2020)]
statsEgy20
```

Out[119…

	country	year	month	dcases				
				mean	std	size	get_ci_lb	get_ci_ub
1224	Egypt	2020	Jan	NaN	NaN	NaN	NaN	NaN
1225	Egypt	2020	Feb	0.062500	0.250000	16.0	-0.070716	0.195716
1226	Egypt	2020	Mar	22.870968	20.228267	31.0	15.451179	30.290756
1227	Egypt	2020	Apr	160.900000	54.996144	30.0	140.364102	181.435898
1228	Egypt	2020	May	627.354839	330.723908	31.0	506.044330	748.665347
1229	Egypt	2020	Jun	1444.200000	176.114853	30.0	1378.437633	1509.962367
1230	Egypt	2020	Jul	831.193548	326.183427	31.0	711.548501	950.838595
1231	Egypt	2020	Aug	156.806452	42.607839	31.0	141.177770	172.435133
1232	Egypt	2020	Sep	141.966667	23.389481	30.0	133.232891	150.700442
1233	Egypt	2020	Oct	140.548387	24.650678	31.0	131.506445	149.590329
1234	Egypt	2020	Nov	278.533333	71.137426	30.0	251.970182	305.096485
1235	Egypt	2020	Dec	714.548387	354.179036	31.0	584.634468	844.462306

In [120…
```
statsEgy20.columns=['country','year','month','mean','std','size','lb','ub']
```

In [121…
```
statsEgy21.columns=['country','year','month','mean','std','size','lb','ub']
```

In [122…
```
x=statsEgy20['month']
x
```

Out[122…
```
1224    Jan
1225    Feb
1226    Mar
1227    Apr
1228    May
1229    Jun
1230    Jul
1231    Aug
1232    Sep
1233    Oct
1234    Nov
1235    Dec
Name: month, dtype: category
Categories (12, object): ['Jan' < 'Feb' < 'Mar' < 'Apr' ... 'Sep' < 'Oct' < 'Nov' < 'Dec']
```

In [123…
```
y1=statsEgy20['mean']
y1
```

Out[123…
```
1224            NaN
1225       0.062500
1226      22.870968
1227     160.900000
1228     627.354839
1229    1444.200000
1230     831.193548
1231     156.806452
1232     141.966667
1233     140.548387
1234     278.533333
1235     714.548387
Name: mean, dtype: float64
```

In [124…
```
y2=statsEgy21['mean']
y2
```

Out[124…
```
1236     899.645161
1237     588.321429
1238     635.709677
1239     847.366667
1240    1132.193548
1241     621.066667
1242      96.129032
1243     134.806452
1244     536.100000
1245     854.612903
1246     918.700000
1247     870.870968
Name: mean, dtype: float64
```

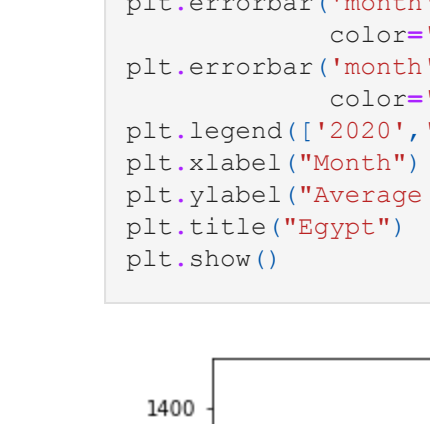
In [125…
```
ci_lb_ub20=[statsEgy20['lb'],statsEgy20['ub']]
err20 = np.abs(ci_lb_ub20 - statsEgy20['mean'].to_numpy())
```

In [126…
```
ci_lb_ub21=[statsEgy21['lb'],statsEgy21['ub']]
err21 = np.abs(ci_lb_ub21 - statsEgy21['mean'].to_numpy())
```

In [127…
```
from matplotlib.transforms import Affine2D
```

In [128…
```
fig, ax = plt.subplots()

trans1 = Affine2D().translate(-0.1, 0.0) + ax.transData
trans2 = Affine2D().translate(+0.1, 0.0) + ax.transData
plt.errorbar('month', 'mean',  yerr=err20, data=statsEgy20,marker='s', capsize=2,
             color='blue', markersize=4, linewidth=1, linestyle='--',transform=trans1)
plt.errorbar('month', 'mean',  yerr=err21, data=statsEgy21,marker='s', capsize=2,
             color='red', markersize=4, linewidth=1, linestyle='--',transform=trans2)
plt.legend(['2020','2021'])
plt.xlabel("Month")
plt.ylabel("Average Daily COVID CASES")
plt.title("Egypt")
plt.show()
```



Assignment 2 part 2 (Discussion & Conclusion)

in this part of the assignment I will be comparing between:-
2020 & 2021
daily cases & daily deaths
egypt & other country

Comparing between 2020 & 2021

In [129…
```
statsEgy=statsdcases[(statsdcases['country']=='Egypt') & (statsdcases['year']==2020)]
statsEgy
```

Out[129…

	country	year	month	dcases				
				mean	std	size	get_ci_lb	get_ci_ub
1224	Egypt	2020	Jan	NaN	NaN	NaN	NaN	NaN
1225	Egypt	2020	Feb	0.062500	0.250000	16.0	-0.070716	0.195716
1226	Egypt	2020	Mar	22.870968	20.228267	31.0	15.451179	30.290756
1227	Egypt	2020	Apr	160.900000	54.996144	30.0	140.364102	181.435898
1228	Egypt	2020	May	627.354839	330.723908	31.0	506.044330	748.665347
1229	Egypt	2020	Jun	1444.200000	176.114853	30.0	1378.437633	1509.962367
1230	Egypt	2020	Jul	831.193548	326.183427	31.0	711.548501	950.838595
1231	Egypt	2020	Aug	156.806452	42.607839	31.0	141.177770	172.435133
1232	Egypt	2020	Sep	141.966667	23.389481	30.0	133.232891	150.700442
1233	Egypt	2020	Oct	140.548387	24.650678	31.0	131.506445	149.590329
1234	Egypt	2020	Nov	278.533333	71.137426	30.0	251.970182	305.096485
1235	Egypt	2020	Dec	714.548387	354.179036	31.0	584.634468	844.462306

this table shows some data about the cases in egypt in 2020

In [130…
```
statsEgy=statsdcases[(statsdcases['country']=='Egypt') & (statsdcases['year']==2021)]
statsEgy
```

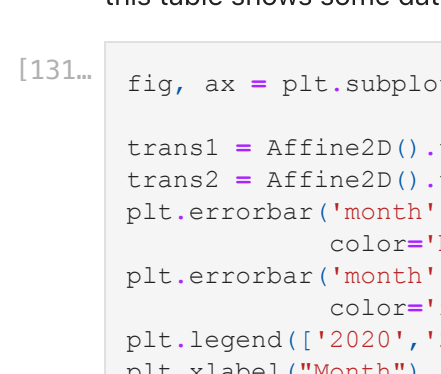
Out[130…

	country	year	month	dcases				
				mean	std	size	get_ci_lb	get_ci_ub
1236	Egypt	2021	Jan	899.645161	251.335970	31.0	807.454383	991.835939
1237	Egypt	2021	Feb	588.321429	40.989659	28.0	572.427298	604.215559
1238	Egypt	2021	Mar	635.709677	36.309038	31.0	622.391415	649.027940
1239	Egypt	2021	Apr	847.366667	92.416405	30.0	812.857814	881.875520
1240	Egypt	2021	May	1132.193548	55.087457	31.0	1111.987306	1152.399791
1241	Egypt	2021	Jun	621.066667	197.770841	30.0	547.217821	694.915513
1242	Egypt	2021	Jul	96.129032	61.659139	31.0	73.512277	118.745787
1243	Egypt	2021	Aug	134.806452	73.635326	31.0	107.796796	161.816107
1244	Egypt	2021	Sep	536.100000	151.276488	30.0	479.612431	592.587569
1245	Egypt	2021	Oct	854.612903	53.664189	31.0	834.928720	874.297087
1246	Egypt	2021	Nov	918.700000	248.217528	30.0	826.014052	1011.385948
1247	Egypt	2021	Dec	870.870968	212.967093	31.0	792.754007	948.987928

this table shows some data about the cases in egypt in 2021

In [131…
```
fig, ax = plt.subplots()

trans1 = Affine2D().translate(-0.1, 0.0) + ax.transData
trans2 = Affine2D().translate(+0.1, 0.0) + ax.transData
plt.errorbar('month', 'mean',  yerr=err20, data=statsEgy20,marker='s', capsize=2,
             color='blue', markersize=4, linewidth=1, linestyle='--',transform=trans1)
plt.errorbar('month', 'mean',  yerr=err21, data=statsEgy21,marker='s', capsize=2,
             color='red', markersize=4, linewidth=1, linestyle='--',transform=trans2)
plt.legend(['2020','2021'])
plt.xlabel("Month")
plt.ylabel("Average Daily COVID CASES")
plt.title("Egypt")
plt.show()
```



This graph shows the difference between 2020 & 2021 according to the average daily covid cases in egypt

By observing we can't have a specific conclusion, yet 2020 had slightly a less number of cases due to the date of the beginning of the virus

Ratio between daily cases & daily deaths

In [132…
```
dfegy=df[df['country']=='Egypt']
stats=dfegy.groupby("month").agg({"dcases": [np.mean, np.std, np.size]})
stats
```

Out[132…

	dcases		
	mean	std	size
month			
Jan	899.645161	251.335970	31
Feb	374.409091	288.087868	44
Mar	329.290323	310.292849	62
Apr	504.133333	354.246465	60
May	879.774194	346.475245	62
Jun	1032.633333	454.674216	60
Jul	463.661290	437.595552	62
Aug	145.806452	60.683224	62
Sep	339.033333	225.855584	60
Oct	497.580645	362.321547	62
Nov	598.616667	370.082496	60
Dec	792.709677	300.346870	62

That table is to show some stats about the daily cases in egypt by month

In [133…
```
dfegy=df[df['country']=='Egypt']
stats=dfegy.groupby("month").agg({"ddeaths": [np.mean, np.std, np.size]})
stats
```

Out[133…

	ddeaths		
	mean	std	size
month			
Jan	54.354839	3.638208	31
Feb	31.181818	24.163467	44
Mar	21.822581	20.717083	62
Apr	28.166667	17.826058	60
May	37.483871	20.947821	62
Jun	51.116667	23.279504	60
Jul	35.596774	27.745010	62
Aug	13.354839	7.503216	62
Sep	18.400000	7.962199	60
Oct	26.709677	16.823468	62
Nov	36.783333	27.679211	60
Dec	36.435484	15.919491	62

That table is to show some stats about the daily deaths in egypt by month

The reason I am presenting these two graphs is to shows the ration between the daily cases and deaths.
deaths:390 cases:6859
the ratio between the daily deaths and cases will nearly be for every 18 cases there is 1 death

Comparing between Egypt & Belgium

In [134…
```
statsBel=statsdcases[(statsdcases['country']=='Belgium') & (statsdcases['year']==2021)]
statsBel
```

Out[134…

	country	year	month	dcases				
				mean	std	size	get_ci_lb	get_ci_ub
396	Belgium	2021	Jan	2053.451613	710.931447	31.0	1792.679853	2314.223373
397	Belgium	2021	Feb	2191.357143	678.358652	28.0	1928.317118	2454.397168
398	Belgium	2021	Mar	3578.774194	1881.668315	31.0	2888.572683	4268.975704
399	Belgium	2021	Apr	3592.533333	1098.126076	30.0	3182.486318	4002.580349
400	Belgium	2021	May	2315.225806	937.140783	31.0	1971.479792	2658.971821
401	Belgium	2021	Jun	771.000000	485.704394	30.0	589.634999	952.365001
402	Belgium	2021	Jul	1276.903226	1219.220526	31.0	829.689526	1724.116925
403	Belgium	2021	Aug	1933.838710	1597.149525	31.0	1347.999534	2519.677885
404	Belgium	2021	Sep	2009.666667	1520.628553	30.0	1441.854633	2577.478700
405	Belgium	2021	Oct	3732.129032	3825.571206	31.0	2328.898185	5135.359879
406	Belgium	2021	Nov	13512.833333	13651.767570	30.0	8415.179545	18610.487122
407	Belgium	2021	Dec	10945.419355	9804.831553	31.0	7348.978106	14541.860604

this table represents Belgium's cases in 2021

In [135…
```
statsEgy=statsdcases[(statsdcases['country']=='Egypt') & (statsdcases['year']==2021)]
statsEgy
```

Out[135…

	country	year	month	dcases				
				mean	std	size	get_ci_lb	get_ci_ub
1236	Egypt	2021	Jan	899.645161	251.335970	31.0	807.454383	991.835939
1237	Egypt	2021	Feb	588.321429	40.989659	28.0	572.427298	604.215559
1238	Egypt	2021	Mar	635.709677	36.309038	31.0	622.391415	649.027940
1239	Egypt	2021	Apr	847.366667	92.416405	30.0	812.857814	881.875520
1240	Egypt	2021	May	1132.193548	55.087457	31.0	1111.987306	1152.399791
1241	Egypt	2021	Jun	621.066667	197.770841	30.0	547.217821	694.915513
1242	Egypt	2021	Jul	96.129032	61.659139	31.0	73.512277	118.745787
1243	Egypt	2021	Aug	134.806452	73.635326	31.0	107.796796	161.816107
1244	Egypt	2021	Sep	536.100000	151.276488	30.0	479.612431	592.587569
1245	Egypt	2021	Oct	854.612903	53.664189	31.0	834.928720	874.297087
1246	Egypt	2021	Nov	918.700000	248.217528	30.0	826.014052	1011.385948
1247	Egypt	2021	Dec	870.870968	212.967093	31.0	792.754007	948.987928

this table represents Egypt's cases in 2021

In [136…
```
dfbel=df[df['country']=='Belgium']
stats=dfbel.groupby("month").agg({"ddeaths": [np.mean, np.std, np.size]})
stats
```

Out[136…

	ddeaths		
	mean	std	size
month			
Jan	50.451613	13.490339	31
Feb	18.240741	19.263746	54
Mar	26.516129	31.279670	62
Apr	135.050000	116.969129	60
May	41.903226	43.412135	62
Jun	8.300000	6.040260	60
Jul	2.612903	2.663382	62
Aug	5.000000	4.537639	62
Sep	5.716667	5.502208	60
Oct	32.274194	39.255852	62
Nov	100.683333	81.143416	60
Dec	67.725806	44.380682	62

In [137…
```
dfegy=df[df['country']=='Egypt']
stats=dfegy.groupby("month").agg({"ddeaths": [np.mean, np.std, np.size]})
stats
```

Out[137…

	ddeaths		
	mean	std	size
month			
Jan	54.354839	3.638208	31
Feb	31.181818	24.163467	44
Mar	21.822581	20.717083	62
Apr	28.166667	17.826058	60
May	37.483871	20.947821	62
Jun	51.116667	23.279504	60
Jul	35.596774	27.745010	62
Aug	13.354839	7.503216	62
Sep	18.400000	7.962199	60
Oct	26.709677	16.823468	62
Nov	36.783333	27.679211	60
Dec	36.435484	15.919491	62

As shown, Egypt had less cases than Belgium and that could mean one of three things:-
Egypt is doing better in controlling the virus
Europe has more of this virus (based on the region)
Egypt is not collecting enough data and that is the most reliable option because of the difference in the population
However, Egypt always had more death numbers and that means that the health organization in egypt is bad; by this information we know that egypt is not doing better in controlling the virus

Thank You!