Fixed Assignment 2

April 10, 2022

1 ASSIGNMENT 2

Author: Mostafa Abdelazim, ID: 900203676

Abstract: The code aims analyze the confidence intervals for 2 data sets. The first one is for matches, and their win/lose probability for different reasons, like home/away or friendly/official matches. The second data set is for reported covid cases in all countries in 20202 and 2021. In both data sets, all the analyzed data are given and graphed throughout the code.

2 Part 1

[1]:
```
import numpy as np
import pandas as pd
import scipy
import scipy.stats
from scipy.stats import norm,t
import statsmodels.api as sm
from statsmodels.stats.proportion import proportion_confint
import matplotlib.pyplot as plt
from pandas.api.types import CategoricalDtype
```

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

[2]:

	date	home_team	away_team	home_score	away_score	\
0	1872-11-30	Scotland	England	0	0	
1	1873-03-08	England	Scotland	4	2	
2	1874-03-07	Scotland	England	2	1	
3	1875-03-06	England	Scotland	2	2	
4	1876-03-04	Scotland	England	3	0	
…	…	…	…	…	…	
43183	2/1/2022	Suriname	Guyana	2	1	
43184	2/2/2022	Burkina Faso	Senegal	1	3	
43185	2/3/2022	Cameroon	Egypt	0	0	
43186	2/5/2022	Cameroon	Burkina Faso	3	3	
43187	2/6/2022	Senegal	Egypt	0	0	

	tournament	city	country	neutral

```
0                    Friendly     Glasgow  Scotland    False
1                    Friendly      London   England    False
2                    Friendly     Glasgow  Scotland    False
3                    Friendly      London   England    False
4                    Friendly     Glasgow  Scotland    False
…                       …          …       …       …
43183                Friendly  Paramaribo  Suriname    False
43184  African Cup of Nations     Yaoundé  Cameroon     True
43185  African Cup of Nations     Yaoundé  Cameroon    False
43186  African Cup of Nations     Yaoundé  Cameroon    False
43187  African Cup of Nations     Yaoundé  Cameroon     True

[43188 rows x 9 columns]
```

[3]:
```
x=df['home_score']-df['away_score']
conditions = [
    (x<0),
    (x>0),
    (x==0)
    ]
```

[4]:
```
values= ['win','lose','draw']
```

[5]:
```
df['result'] = np.select(conditions, values)
```

[6]:
```
x=df['result'].value_counts()
```

[7]:
```
x=np.array(x)
```

[8]:
```
x
```

[8]:
```
array([21009, 12224,  9955], dtype=int64)
```

[9]:
```
conditions = [
    (df['tournament']=='Friendly'),
    (df['tournament']!='Friendly')
    ]
```

[10]:
```
values=['Friendly','Official']
```

[11]:
```
df['typematch'] = np.select(conditions, values)
```

[12]:
```
x=pd.crosstab(df['typematch'],df['result'],margins=True)
x
```

[12]:
```
result     draw   lose    win    All
typematch
Friendly   4329   8141   4806  17276
```

```
Official   5626  12868   7418  25912
All        9955  21009  12224  43188
```

[13]:
```
x=np.array(x)
x
```

[13]:
```
array([[ 4329,  8141,  4806, 17276],
       [ 5626, 12868,  7418, 25912],
       [ 9955, 21009, 12224, 43188]], dtype=int64)
```

[14]:
```
CI_win_friendly=proportion_confint(count=x[0,2],nobs=x[0,3],alpha=(1-.95))
CI_win_friendly
```

[14]:
```
(0.27150736589666685, 0.2848714254902283)
```

[15]:
```
CI_win_official=proportion_confint(count=x[1,2],nobs=x[1,3],alpha=(1-.95))
CI_win_official
```

[15]:
```
(0.2807729187467606, 0.29178033843138085)
```

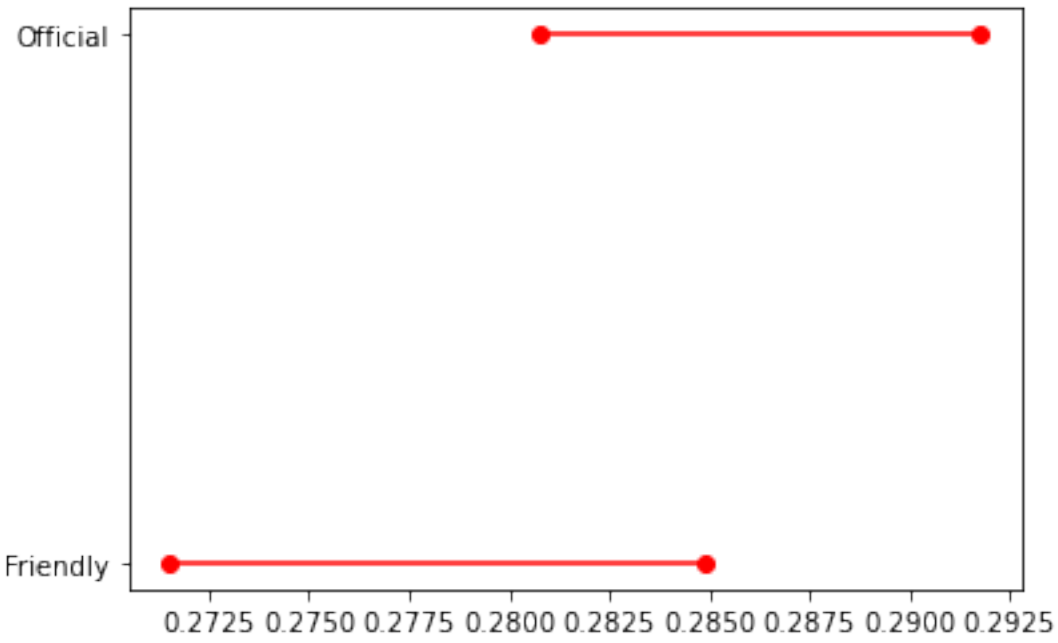
[16]:
```
ci_win = {}
ci_win['Typematch'] = ['Friendly','Official']
ci_win['lb'] = [CI_win_friendly[0],CI_win_official[0]]
ci_win['ub'] = [CI_win_friendly[1],CI_win_official[1]]
df_ci3= pd.DataFrame(ci_win)
df_ci3
```

[16]:

	Typematch	lb	ub
0	Friendly	0.271507	0.284871
1	Official	0.280773	0.291780

[17]:
```
for lb,ub,y in zip(df_ci3['lb'],df_ci3['ub'],range(len(df_ci3))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(df_ci3)),list(df_ci3['Typematch']))
```

[17]:
```
([<matplotlib.axis.YTick at 0x1f0322f06a0>,
  <matplotlib.axis.YTick at 0x1f0322d9ee0>],
 [Text(0, 0, 'Friendly'), Text(0, 1, 'Official')])
```



[18]:
```
CI_lose_friendly=proportion_confint(count=x[0,1],nobs=x[0,3],alpha=(1-.95))
CI_lose_friendly
```

[18]: (0.46378827932197364, 0.47867525390331)

[19]:
```
CI_lose_official=proportion_confint(count=x[1,1],nobs=x[1,3],alpha=(1-.95))
CI_lose_official
```

[19]: (0.4905161288707065, 0.5026916513083611)

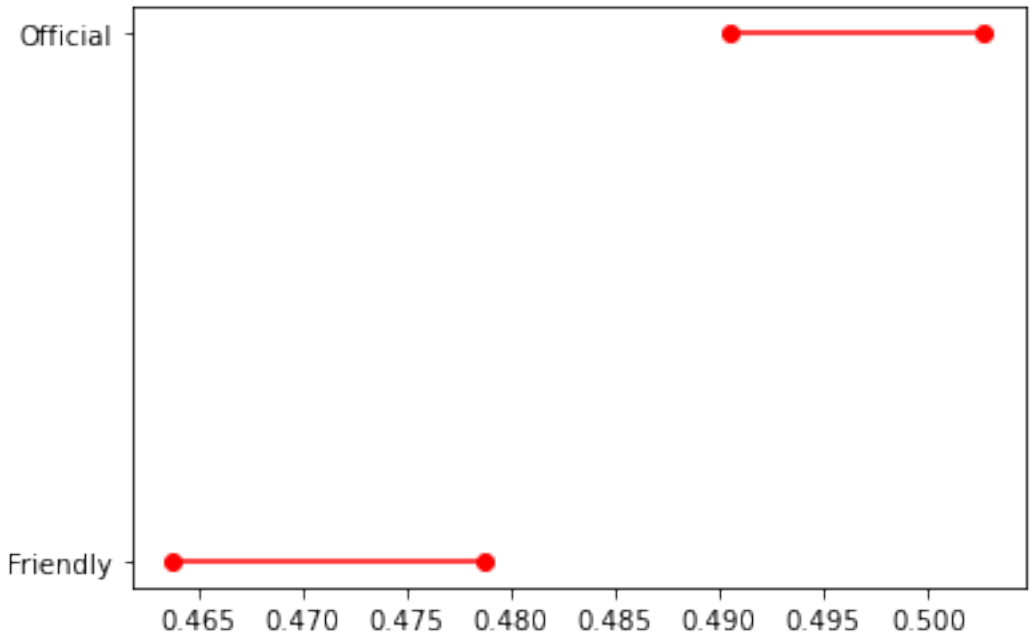
[20]:
```
ci_lose = {}
ci_lose['Typematch'] = ['Friendly','Official']
ci_lose['lb'] = [CI_lose_friendly[0],CI_lose_official[0]]
ci_lose['ub'] = [CI_lose_friendly[1],CI_lose_official[1]]
df_ci4= pd.DataFrame(ci_lose)
df_ci4
```

[20]:

	Typematch	lb	ub
0	Friendly	0.463788	0.478675
1	Official	0.490516	0.502692

[21]:
```
for lb,ub,y in zip(df_ci4['lb'],df_ci4['ub'],range(len(df_ci4))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(df_ci4)),list(df_ci4['Typematch']))
```

[21]:
```
([<matplotlib.axis.YTick at 0x1f03434ca90>,
  <matplotlib.axis.YTick at 0x1f03434c310>],
 [Text(0, 0, 'Friendly'), Text(0, 1, 'Official')])
```



[22]:
```
df['country'].value_counts()
```

[22]:
```
United States          1237
France                  818
Malaysia                744
England                 717
Sweden                  655
                       …
Belgian Congo             1
Portuguese Guinea         1
Bohemia and Moravia       1
Lautoka                   1
Mali Federation           1
Name: country, Length: 267, dtype: int64
```

[23]:
```
dfus=df[df['country']=='United States']
```

[24]:
```
conditions = [
    (dfus['tournament']=='Friendly'),
    (dfus['tournament']!='Friendly')
    ]
```

[25]:
```
values=['Friendly','Official']
```

[26]:
```
dfus['Typematch'] = np.select(conditions, values)
```

```
C:\Users\lenovo\AppData\Local\Temp/ipykernel_2896/327193532.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
  dfus['Typematch'] = np.select(conditions, values)
```

[27]:
```
x=pd.crosstab(dfus['typematch'],dfus['result'],margins=True)
x
```

[27]:

result	draw	lose	win	All
typematch				
Friendly	180	247	226	653
Official	114	315	155	584
All	294	562	381	1237

[28]:
```
x=np.array(x)
x
```

[28]:
```
array([[ 180,  247,  226,  653],
       [ 114,  315,  155,  584],
       [ 294,  562,  381, 1237]], dtype=int64)
```

[29]:
```
CI_uswin_friendly=proportion_confint(count=x[1,2],nobs=x[1,3],alpha=(1-.95))
CI_uswin_friendly
```

[29]:
```
(0.22959939266607973, 0.30122252514213943)
```

[30]:
```
CI_uswin_official=proportion_confint(count=x[0,2],nobs=x[0,3],alpha=(1-.95))
CI_uswin_official
```

[30]:
```
(0.3096072474351973, 0.38258264536725295)
```

[31]:
```
ci_uswin = {}
ci_uswin['Typematch'] = ['Friendly','Official']
ci_uswin['lb'] = [CI_uswin_friendly[0],CI_uswin_official[0]]
ci_uswin['ub'] = [CI_uswin_friendly[1],CI_uswin_official[1]]
df_cius= pd.DataFrame(ci_uswin)
df_cius
```

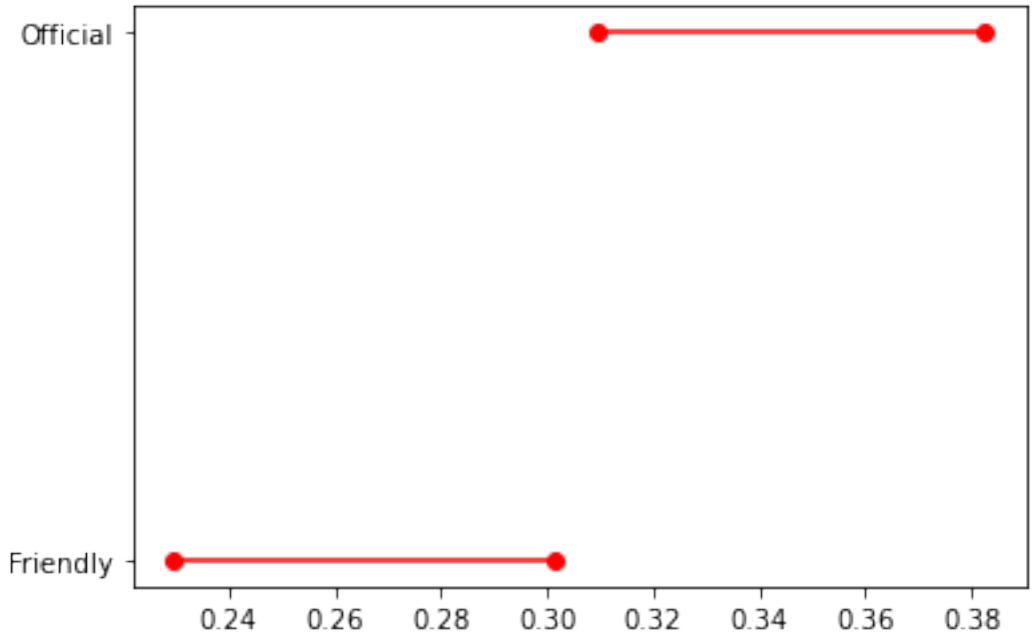
[31]:

	Typematch	lb	ub
0	Friendly	0.229599	0.301223

```
1  Official  0.309607  0.382583
```

[32]:
```
for lb,ub,y in zip(df_cius['lb'],df_cius['ub'],range(len(df_cius))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(df_cius)),list(df_cius['Typematch']))
```

[32]:
```
([<matplotlib.axis.YTick at 0x1f0327d4280>,
  <matplotlib.axis.YTick at 0x1f0327cfac0>],
 [Text(0, 0, 'Friendly'), Text(0, 1, 'Official')])
```



[33]:
```
CI_uslose_friendly=proportion_confint(count=x[0,1],nobs=x[0,3],alpha=(1-.95))
CI_uslose_friendly
```

[33]:
```
(0.3410587617395715, 0.4154496609250533)
```

[34]:
```
CI_uslose_official=proportion_confint(count=x[1,1],nobs=x[1,3],alpha=(1-.95))
CI_uslose_official
```

[34]:
```
(0.4989576113893054, 0.5798095118983658)
```

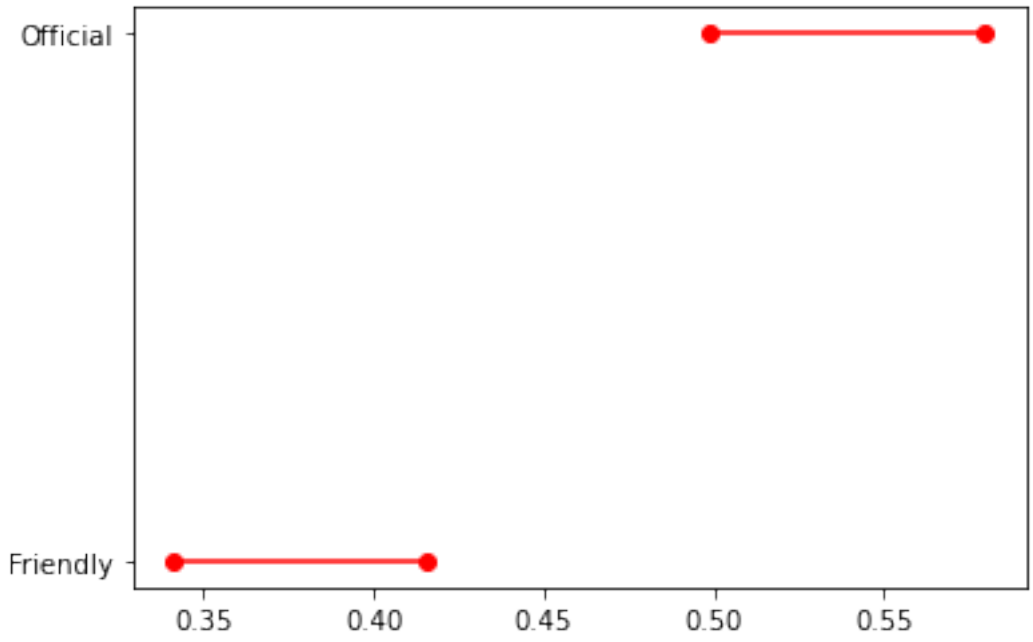
[35]:
```
ci_uslose = {}
ci_uslose['Typematch'] = ['Friendly','Official']
ci_uslose['lb'] = [CI_uslose_friendly[0],CI_uslose_official[0]]
ci_uslose['ub'] = [CI_uslose_friendly[1],CI_uslose_official[1]]
df_cius= pd.DataFrame(ci_uslose)
df_cius
```

[35]:

	Typematch	lb	ub
0	Friendly	0.341059	0.41545
1	Official	0.498958	0.57981

[36]:
```
for lb,ub,y in zip(df_cius['lb'],df_cius['ub'],range(len(df_cius))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(df_cius)),list(df_cius['Typematch']))
```

[36]:
```
([<matplotlib.axis.YTick at 0x1f032839190>,
  <matplotlib.axis.YTick at 0x1f0328319d0>],
 [Text(0, 0, 'Friendly'), Text(0, 1, 'Official')])
```



[37]:
```
dfus['home']=(dfus['home_team']=='United States')
```

```
C:\Users\lenovo\AppData\Local\Temp/ipykernel_2896/3054155073.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
  dfus['home']=(dfus['home_team']=='United States')
```

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

[38]:
```
result  draw  lose  win   All
home
False    203   333  290   826
True      91   229   91   411
All      294   562  381  1237
```

[39]:
```
x=np.array(x)
x
```

[39]:
```
array([[ 203,  333,  290,  826],
       [  91,  229,   91,  411],
       [ 294,  562,  381, 1237]], dtype=int64)
```

[40]:
```
CI_uswin_home=proportion_confint(count=x[1,2],nobs=x[1,3],alpha=(1-.95))
CI_uswin_home
```

[40]:
```
(0.1812708525133201, 0.26155153191490377)
```

[41]:
```
CI_uswin_away=proportion_confint(count=x[0,2],nobs=x[0,3],alpha=(1-.95))
CI_uswin_away
```

[41]:
```
(0.31853895723001224, 0.3836402195254357)
```

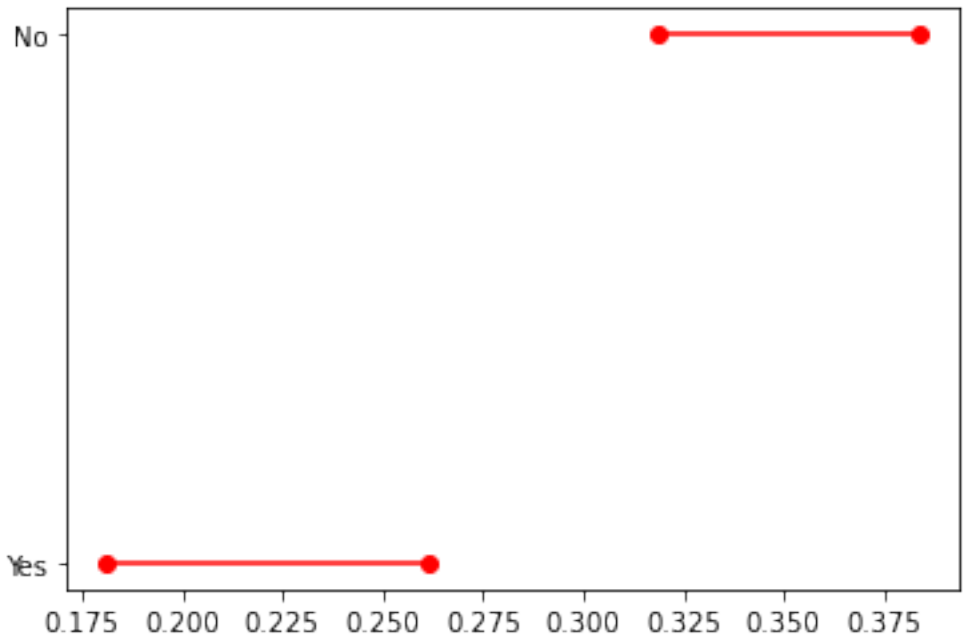
[42]:
```
ci_uswin = {}
ci_uswin['home'] = ['Yes','No']
ci_uswin['lb'] = [CI_uswin_home[0],CI_uswin_away[0]]
ci_uswin['ub'] = [CI_uswin_home[1],CI_uswin_away[1]]
df_ci= pd.DataFrame(ci_uswin)
df_ci
```

[42]:

	home	lb	ub
0	Yes	0.181271	0.261552
1	No	0.318539	0.383640

[43]:
```
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']))
```

[43]:
```
([<matplotlib.axis.YTick at 0x1f03289bcd0>,
  <matplotlib.axis.YTick at 0x1f03289b4f0>],
 [Text(0, 0, 'Yes'), Text(0, 1, 'No')])
```



[44]:
```
CI_uslose_home=proportion_confint(count=x[1,1],nobs=x[1,3],alpha=(1-.95))
CI_uslose_home
```

[44]: (0.5091557759095878, 0.6051994552339645)

[45]:
```
CI_uslose_away=proportion_confint(count=x[0,1],nobs=x[0,3],alpha=(1-.95))
CI_uslose_away
```

[45]: (0.3696955835778047, 0.4365998159379338)

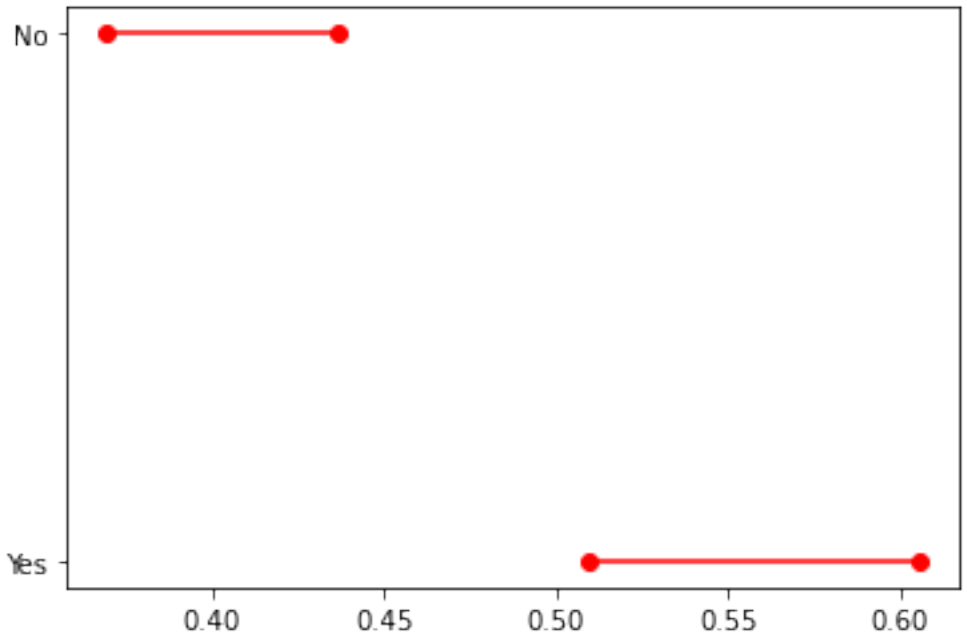
[46]:
```
ci_uslose = {}
ci_uslose['home'] = ['Yes','No']
ci_uslose['lb'] = [CI_uslose_home[0],CI_uslose_away[0]]
ci_uslose['ub'] = [CI_uslose_home[1],CI_uslose_away[1]]
df_ci= pd.DataFrame(ci_uslose)
df_ci
```

[46]:

	home	lb	ub
0	Yes	0.509156	0.605199
1	No	0.369696	0.436600

[47]:
```
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']))
```

```
[47]: ([<matplotlib.axis.YTick at 0x1f0328ff0d0>,
         <matplotlib.axis.YTick at 0x1f032811fd0>],
        [Text(0, 0, 'Yes'), Text(0, 1, 'No')])
```



3 Conclusion for Part 1

The code aims to find the 95% confidence interval for won/lost matches for both match types, Friendly and Official. The first 2 confidence intervals were for won/lost matches in friendly and offical matches for all countries in general without choosing a specific one. The confidence interval for winning in both both matches types were totally different. The range of the interval for winning an official match is less than a friendly match. Which led to a high margin of error for winning a friendly match, and the propability of winning a friendly match is high than winning an official match. Moreover, for losing, it's the same thing. The probability of losing a match is higher in friendly matches and the MOE is also bigger. The first 2 confidence intervals were for won/lost matches in friendly and offical matches for the most common country, United States. The range of the interval for winning an official match is more than a friendly match. Which led to a high margin of error for winning an official match, and the propability of winning an official match is higher than winning a friendly match. Moreover,for losing, it's the same thing. The probability of losing a match is higher in official matches and the MOE is also bigger. I also did the same thing for winning and lossing in both home and away matches

4 Part 2

[48]:
```
df1=pd.read_csv('covid_data.csv',encoding='latin-1')
df1
```

[48]:
```
              date iso3c      country              income  \
0       2020-02-24   AFG  Afghanistan          Low income
1       2020-02-25   AFG  Afghanistan          Low income
2       2020-02-26   AFG  Afghanistan          Low income
3       2020-02-27   AFG  Afghanistan          Low income
4       2020-02-28   AFG  Afghanistan          Low income
…              …   …           …                   …
122838  2021-12-27   ZWE     Zimbabwe  Lower middle income
122839  2021-12-28   ZWE     Zimbabwe  Lower middle income
122840  2021-12-29   ZWE     Zimbabwe  Lower middle income
122841  2021-12-30   ZWE     Zimbabwe  Lower middle income
122842  2021-12-31   ZWE     Zimbabwe  Lower middle income

                    region continent  dcases  ddeaths  population weekdays  \
0               South Asia      Asia       5        0    38041754      Mon
1               South Asia      Asia       0        0    38041754      Tue
2               South Asia      Asia       0        0    38041754      Wed
3               South Asia      Asia       0        0    38041754      Thu
4               South Asia      Asia       0        0    38041754      Fri
…                      …       …     …      …        …         …       …
122838  Sub-Saharan Africa    Africa    1098       17    14645468      Mon
122839  Sub-Saharan Africa    Africa    2099       32    14645468      Tue
122840  Sub-Saharan Africa    Africa       0        0    14645468      Wed
122841  Sub-Saharan Africa    Africa    4180       57    14645468      Thu
122842  Sub-Saharan Africa    Africa    1530        7    14645468      Fri

       month
0        Feb
1        Feb
2        Feb
3        Feb
4        Feb
…      …
122838   Dec
122839   Dec
122840   Dec
122841   Dec
122842   Dec

[122843 rows x 11 columns]
```

[49]:
```
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)
df1['month'] = df1['month'].astype(cat_type)
```

[50]:
```
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


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
```

[51]:
```
df1['date'][0]

df1['date'] = pd. to_datetime(df1['date'],format='%Y-%m-%d')

df1['date'][0]

df1['year'] = pd. DatetimeIndex(df1['date']). year

df1['year'][0]
```

[51]:
```
2020
```

[52]:
```
ratio=df1['dcases']/df1['ddeaths']

df1['ratio']=ratio

df1 = df1.replace([np.inf, -np.inf], np.nan).dropna(axis=0)

df1=df1.reset_index()

df1
```

[52]:
```
        index       date iso3c      country              income  \
0          28 2020-03-23   AFG  Afghanistan          Low income
1          31 2020-03-26   AFG  Afghanistan          Low income
```

```
2          34 2020-03-29   AFG  Afghanistan           Low income
3          39 2020-04-03   AFG  Afghanistan           Low income
4          41 2020-04-05   AFG  Afghanistan           Low income
…         …         …   …          …                    …
67768  122837 2021-12-26   ZWE     Zimbabwe  Lower middle income
67769  122838 2021-12-27   ZWE     Zimbabwe  Lower middle income
67770  122839 2021-12-28   ZWE     Zimbabwe  Lower middle income
67771  122841 2021-12-30   ZWE     Zimbabwe  Lower middle income
67772  122842 2021-12-31   ZWE     Zimbabwe  Lower middle income

                    region continent  dcases  ddeaths  population weekdays  \
0              South Asia      Asia       6        1    38041754      Mon
1              South Asia      Asia       6        1    38041754      Thu
2              South Asia      Asia       8        2    38041754      Sun
3              South Asia      Asia      34        1    38041754      Fri
4              South Asia      Asia      29        2    38041754      Sun
…                     …       …     …       …          …       …
67768  Sub-Saharan Africa    Africa     605        6    14645468      Sun
67769  Sub-Saharan Africa    Africa    1098       17    14645468      Mon
67770  Sub-Saharan Africa    Africa    2099       32    14645468      Tue
67771  Sub-Saharan Africa    Africa    4180       57    14645468      Thu
67772  Sub-Saharan Africa    Africa    1530        7    14645468      Fri

      month  year       ratio
0       Mar  2020    6.000000
1       Mar  2020    6.000000
2       Mar  2020    4.000000
3       Apr  2020   34.000000
4       Apr  2020   14.500000
…     …   …          …
67768   Dec  2021  100.833333
67769   Dec  2021   64.588235
67770   Dec  2021   65.593750
67771   Dec  2021   73.333333
67772   Dec  2021  218.571429

[67773 rows x 14 columns]
```

[53]:
```
cy=df1.groupby(['continent','year']).agg({"ratio": [np.mean, np.std, np.
 ↪size,get_ci_lb,get_ci_ub]})

cy=cy.reset_index()

cy= pd.DataFrame(cy)

cy
```

[53]:

	continent	year	ratio				\
			mean	std	size	get_ci_lb	
0	Africa	2020	52.623736	62.672790	4892	50.867061	
1	Africa	2021	75.063308	206.868302	8449	70.651652	
2	Asia	2020	106.957897	157.389473	7393	103.369632	
3	Asia	2021	126.452824	195.776606	12025	122.953291	
4	Europe	2020	85.172982	158.693265	8597	81.817969	
5	Europe	2021	165.387291	524.748437	11756	155.900613	
6	North America(continent)	2020	58.520703	84.557767	2924	55.454557	
7	North America(continent)	2021	85.684798	126.416057	4606	82.033036	
8	Oceania	2020	39.919436	69.552255	170	29.388766	
9	Oceania	2021	194.977007	366.457359	381	158.062720	
10	South America(continent)	2020	46.966987	56.138845	2644	44.826169	
11	South America(continent)	2021	60.432168	92.791185	3936	57.532416	

	get_ci_ub
0	54.380410
1	79.474964
2	110.546162
3	129.952357
4	88.527996
5	174.873970
6	61.586848
7	89.336561
8	50.450106
9	231.891293
10	49.107805
11	63.331920

[54]:
```
ry=df1.groupby(['region','year']).agg({"ratio": [np.mean, np.std, np.
↪size,get_ci_lb,get_ci_ub]})

ry=ry.reset_index()

ry= pd.DataFrame(ry)

ry
```

[54]:

	region	year	ratio			\
			mean	std	size	
0	East Asia & Pacific	2020	83.567307	154.946570	1798	
1	East Asia & Pacific	2021	158.944043	266.651870	3834	
2	Europe & Central Asia	2020	85.533757	155.144980	9958	
3	Europe & Central Asia	2021	154.876803	485.184235	13977	
4	Latin America & Caribbean	2020	51.769371	74.371591	4970	
5	Latin America & Caribbean	2021	68.812508	106.545737	7815	

```
6   Middle East & North Africa  2020  110.395171  162.071844   4225
7   Middle East & North Africa  2021  117.932518  181.088283   5980
8        North America(region)  2020   63.547620   55.760176    598
9        North America(region)  2021  130.337613  155.721731    727
10                  South Asia  2020   84.544853   93.638844   1375
11                  South Asia  2021   73.658405   80.526434   2161
12          Sub-Saharan Africa  2020   54.795871   67.315740   3696
13          Sub-Saharan Africa  2021   80.322500  227.180471   6659
```

```
     get_ci_lb   get_ci_ub
0    76.400465   90.734149
1   150.500906  167.387180
2    82.486195   88.581319
3   146.832556  162.921050
4    49.701217   53.837524
5    66.449926   71.175090
6   105.506770  115.283571
7   113.341851  122.523185
8    59.069425   68.025815
9   118.999136  141.676090
10   79.591086   89.498619
11   70.261350   77.055460
12   52.624965   56.966777
13   74.865002   85.779997
```

[55]:
```
iy=df1.groupby(['income','year']).agg({"ratio": [np.mean, np.std, np.
 ↪size,get_ci_lb,get_ci_ub]})

iy=iy.reset_index()

iy= pd.DataFrame(iy)

iy
```

[55]:
```
                income  year       ratio                                \
                                    mean         std   size   get_ci_lb
0          High income  2020  110.071092  184.041271   9477  106.365280
1          High income  2021  198.575163  510.705591  13416  189.932524
2           Low income  2020   51.368931   75.219109   2388   48.350513
3           Low income  2021   55.006252  105.314757   4058   51.765014
4  Lower middle income  2020   59.430011   65.567399   5894   57.755761
5  Lower middle income  2021   79.194789  158.801813   9668   76.028944
6  Upper middle income  2020   63.920043   99.381190   8861   61.850520
7  Upper middle income  2021   82.289241  159.718291  14011   79.644363
```

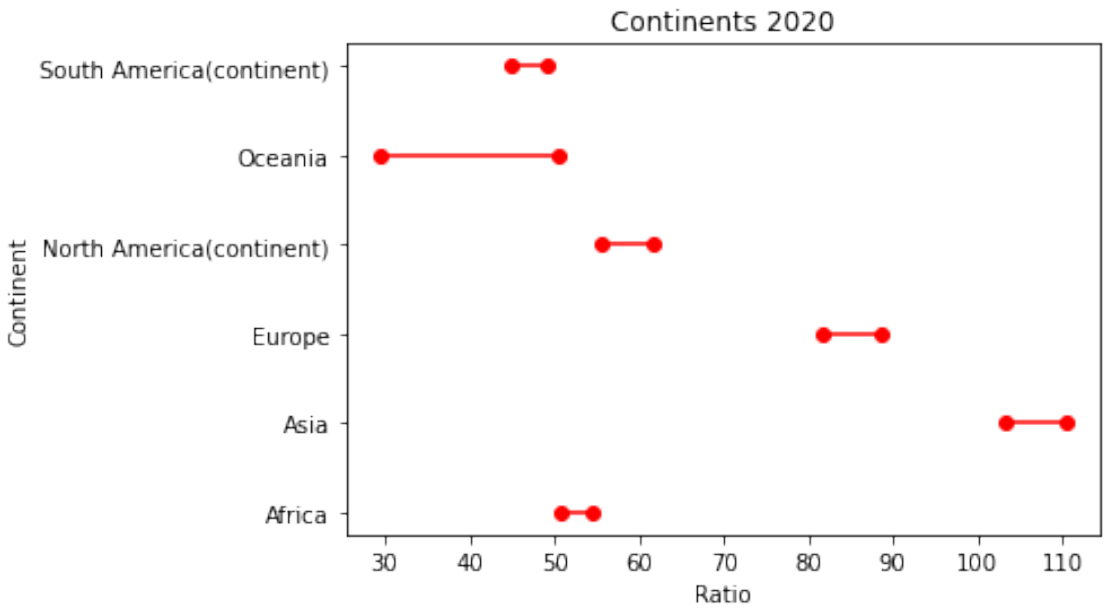
```
    get_ci_ub
0  113.776903
1  207.217802
2   54.387349
3   58.247489
4   61.104260
5   82.360633
6   65.989566
7   84.934119
```

[56]:
```
cy20=cy[(cy['year']==2020)]
cy21=cy[(cy['year']==2021)]

cy20.columns
cy20.columns=['continent','year','mean','std','size','get_ci_lb','get_ci_ub']
cy21.columns
cy21.columns=['continent','year','mean','std','size','get_ci_lb','get_ci_ub']
```

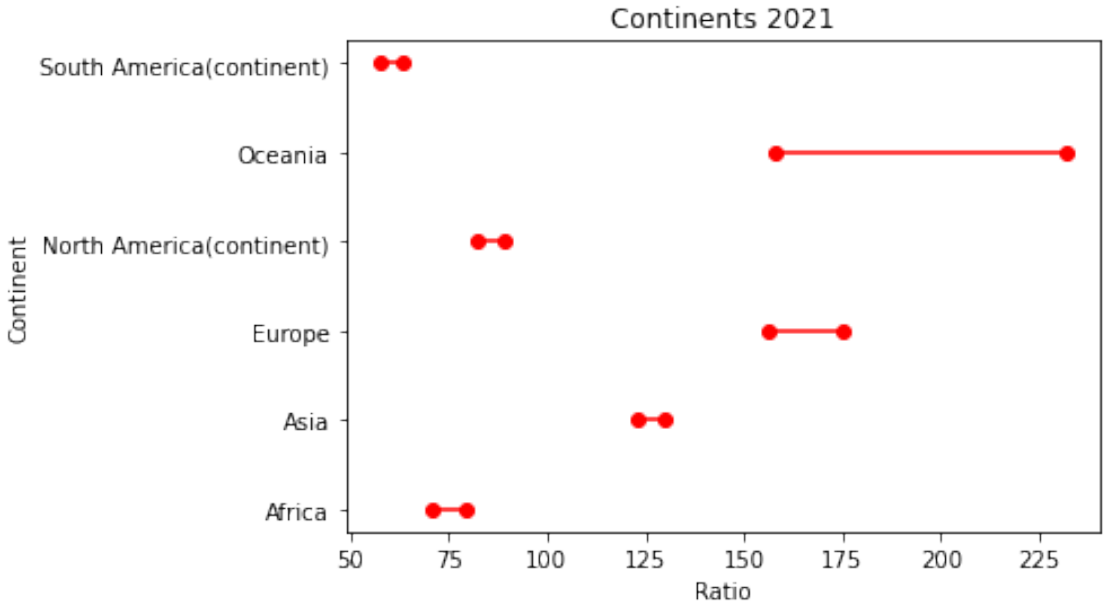
[57]:
```
import matplotlib.pyplot as plt
for lb,ub,y in zip(cy20['get_ci_lb'],cy20['get_ci_ub'],range(len(cy))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(cy20)),list(cy20['continent']))
plt.xlabel("Ratio")
plt.ylabel("Continent")
plt.title("Continents 2020")
```

[57]:
```
Text(0.5, 1.0, 'Continents 2020')
```



[58]:
```
import matplotlib.pyplot as plt
for lb,ub,y in zip(cy21['get_ci_lb'],cy21['get_ci_ub'],range(len(cy))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(cy21)),list(cy21['continent']))
plt.xlabel("Ratio")
plt.ylabel("Continent")
plt.title("Continents 2021")
```

[58]:
```
Text(0.5, 1.0, 'Continents 2021')
```

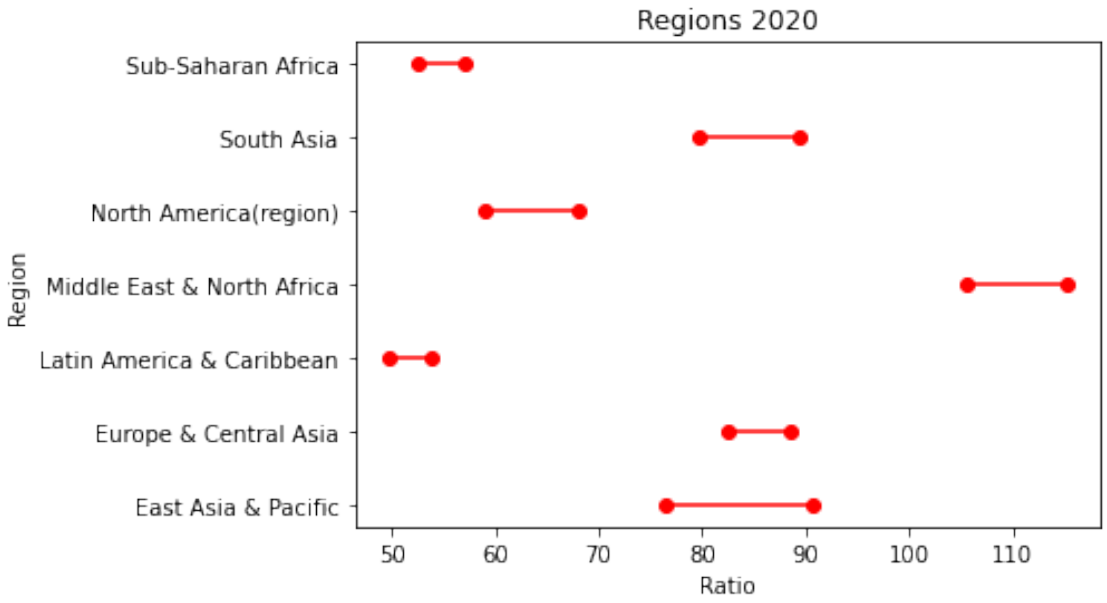


[59]:
```
ry20=ry[(ry['year']==2020)]
ry21=ry[(ry['year']==2021)]

ry20.columns
ry20.columns=['continent','year','mean','std','size','get_ci_lb','get_ci_ub']
ry21.columns
ry21.columns=['continent','year','mean','std','size','get_ci_lb','get_ci_ub']
```

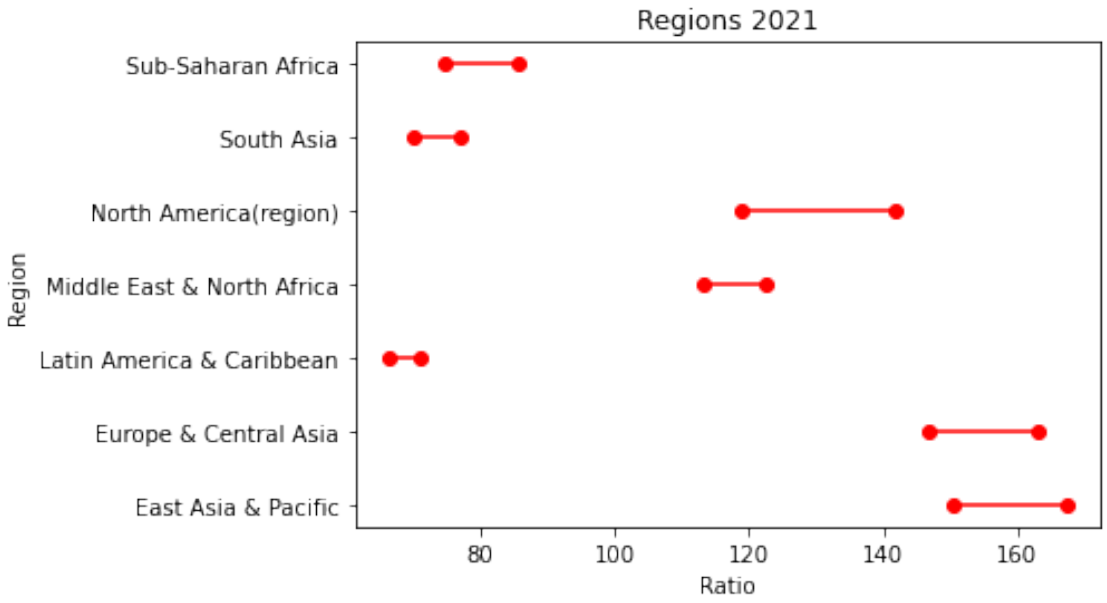
[60]:
```
import matplotlib.pyplot as plt
for lb,ub,y in zip(ry20['get_ci_lb'],ry20['get_ci_ub'],range(len(ry20))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(ry20)),list(ry20['continent']))
plt.xlabel("Ratio")
plt.ylabel("Region")
plt.title("Regions 2020")
```

[60]:
```
Text(0.5, 1.0, 'Regions 2020')
```



[61]:
```
import matplotlib.pyplot as plt
for lb,ub,y in zip(ry21['get_ci_lb'],ry21['get_ci_ub'],range(len(ry21))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(ry21)),list(ry21['continent']))
plt.xlabel("Ratio")
plt.ylabel("Region")
plt.title("Regions 2021")
```

[61]: Text(0.5, 1.0, 'Regions 2021')

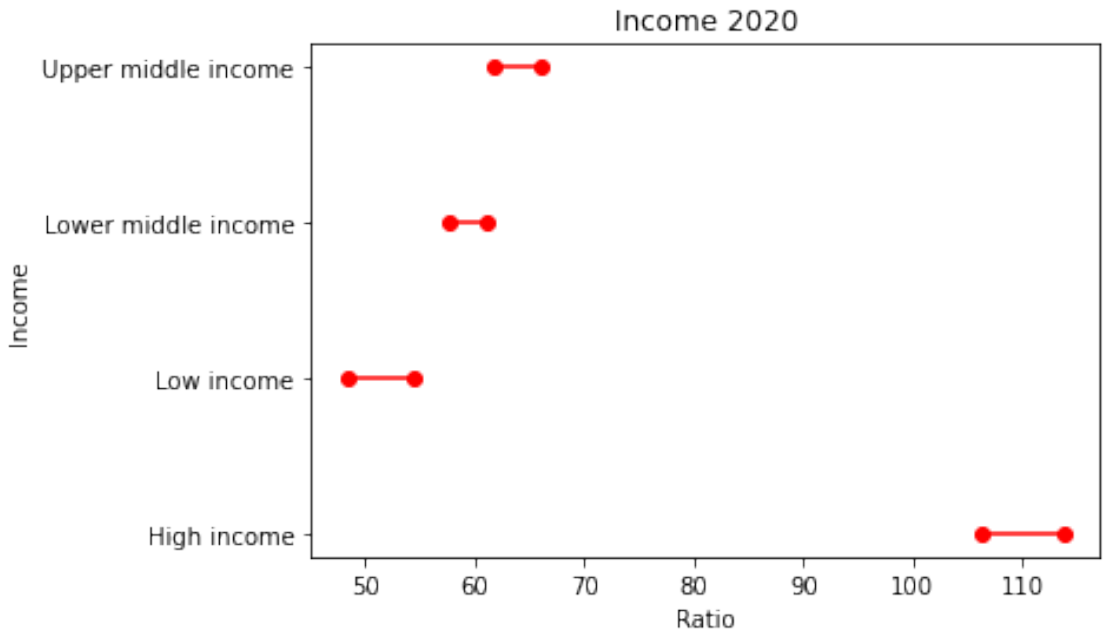


[62]:
```
iy20=iy[(iy['year']==2020)]
iy21=iy[(iy['year']==2021)]

iy20.columns
iy20.columns=['continent','year','mean','std','size','get_ci_lb','get_ci_ub']
iy21.columns
iy21.columns=['continent','year','mean','std','size','get_ci_lb','get_ci_ub']
```

[63]:
```
import matplotlib.pyplot as plt
for lb,ub,y in zip(iy20['get_ci_lb'],iy20['get_ci_ub'],range(len(iy20))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(iy20)),list(iy20['continent']))
plt.xlabel("Ratio")
plt.ylabel("Income")
plt.title("Income 2020")
```

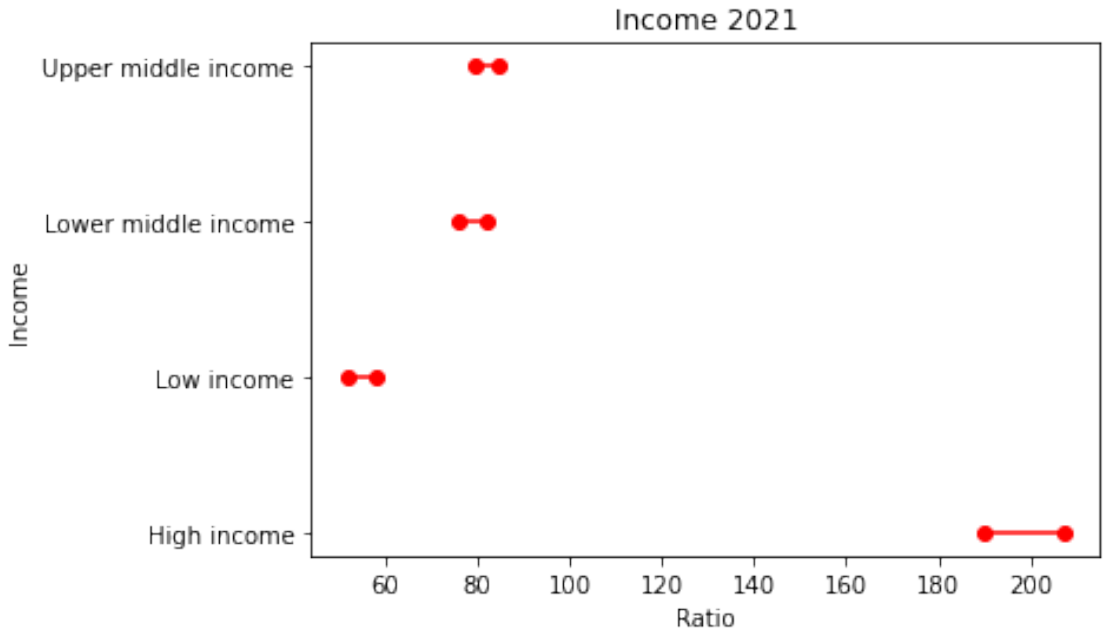
[63]:
```
Text(0.5, 1.0, 'Income 2020')
```



[64]:
```
import matplotlib.pyplot as plt
for lb,ub,y in zip(iy21['get_ci_lb'],iy21['get_ci_ub'],range(len(iy21))):
    plt.plot((lb,ub),(y,y),'ro-')
plt.yticks(range(len(iy21)),list(iy21['continent']))
plt.xlabel("Ratio")
```

```
plt.ylabel("Income")
plt.title("Income 2021")
```

[64]: `Text(0.5, 1.0, 'Income 2021')`



5 Conclusion for Part 2

This code aims to analyze the ratio between cases and deaths in 2020 and 2021, and compare them with respect to 3 aspects, continent, region, and income. Throughout all the aspects, the ratio in 2020 is below 120 for all parts in the graph. While in 2021, the ratio increases in all aspects and the parts within them.