

KNOWLEDGE OF DATA

The data used in this research has been collected from the GapMinder Dataset. GapMinder has collected data from 192 United Nations members and 24 from other areas. Data from a total of 215 countries has been generated in this dataset.

- The original link to the excel file is: [Gapminder.csv](#)
- The codebook for this dataset can be found at [GapminderCodebook.pdf](#).

This data is itself a compilation of various datasets from different sources including the US Census Bureau's International Database, Institute for Health Metrics and Evaluation, World Bank and United Nations Statistics Division. The clubbed data from 215 countries from all over the world has been represented in a single spreadsheet. Therefore, the unique identifier in this data is Country name.

The data has been classified under 15 variables including economical, health, living standards and environmental parameters. A detailed description of all 15 variables can be accessed from the links mentioned above.

For our research, we have picked 2 variables from the data set that represent the Employment rate and Female employment rate of the respective countries.

Technically in our dataset the Employment rate have been listed under the variable name "employrate" and Female employment rate have been listed under "femaleemployrate".

The third variable taken is, Income per person which will predict some information of income per person in different countries. Technically in our dataset the Incomes have been listed under the variable name "incomeperperson".

Also, the country name being the unique identifier for both our indicators is technically mentioned as "country".

In our entire research we will take these three variables (incomeperperson, employrate and female employrate) together and study them to obtain a relation (if any) between the employment rate, female employment rate and income per person country wise.

Discussion of some data related to Employment rate and Female Employment rate:

The data is taken from Gapminder dataset. There is a tremendous diversity in the level and time-series pattern of the self-employment rate across countries. After documenting this fact with cross-section and time-series data on industrialized and lesser-developed countries, this paper presents and tests a series of hypotheses concerning the sources of this diversity. We show that the major explanation for this diversity is the stage of economic development. Although economic development is an extremely powerful force behind the secular decline in self-employment rates, the convergence of several factors in the 1970s tended to stem the secular decline in the self-employment rate for many countries. Of the 23 OECD countries we examined, 15 had increases in the self-employment rate during the 1970s or 1980s. It is likely, however, that these factors are temporary and that self-employment will continue its downwards trend as per-capita wealth increases in the developed and developing world.

The overall trend in self-employment, at the economy level in the years since 1966, has been down in most countries. The main exceptions to this are Portugal, New Zealand and the United Kingdom where the trend has been upward. For most countries there is a negative relationship between the self-employment rate and the unemployment rate. The probability of being self-employed is higher among men than women and rises with age. The least educated have the highest probability of being self-employed, however, evidence is found that the most highly educated also have relatively high probabilities. The self-employed have higher levels of job satisfaction than employees. I could find no evidence that increases in the self-employment rate increased the real growth rate of the economy; in fact there was even evidence of the opposite. The self-employed are less willing to move from their neighborhoods, towns and regions than are employees, presumably because of the pull of their customers. I developed a flexibility index based on information provided by individuals in 1995. According to this index the US economy was the most flexible, followed by Canada, Germany and the Netherlands. Latvia, Russia and Hungary were found to be the least flexible countries. Of the OECD countries examined, Austria and Ireland were ranked lowest.

Comparison of self-employment between India and OECD countries:

More than 60% countries have higher employment rate than India, Australia has 61.5% employment rate, Austria has 57.09%, Belgium has 48.59%, Canada has 63.5%, Chile has 51%, US has 62.29% and other countries have also employment rate in this range while India has 55.41% employment rate.

Among all OECD countries, the highest employment rate is found in Iceland which is 73.59% which is much higher as compare to India. They all are over 15 yrs. old people. In recent years, Iceland has 75.4% employment rate in 2018 and 75.8% employment rate in 2017.

Iceland has the highest employment rate for over 15 yrs. old people among all OECD countries and India.

Highest Employment Rates for Ages 15-24

In recent years, Iceland has 89.1% employment rate in 2018 and 89.8% employment rate in 2017.

After taking a quick look at both of the tables showing the employment rates for the two different age groups, one can quickly realize that the world's employment rate is way higher for individuals aged 25-54 than it is for those aged 15-24. What is more, with the exception of Iceland, which is the country with the highest employment rate among this age rate too (89,1% of the country's 25-54-year-olds were employed in 2018)

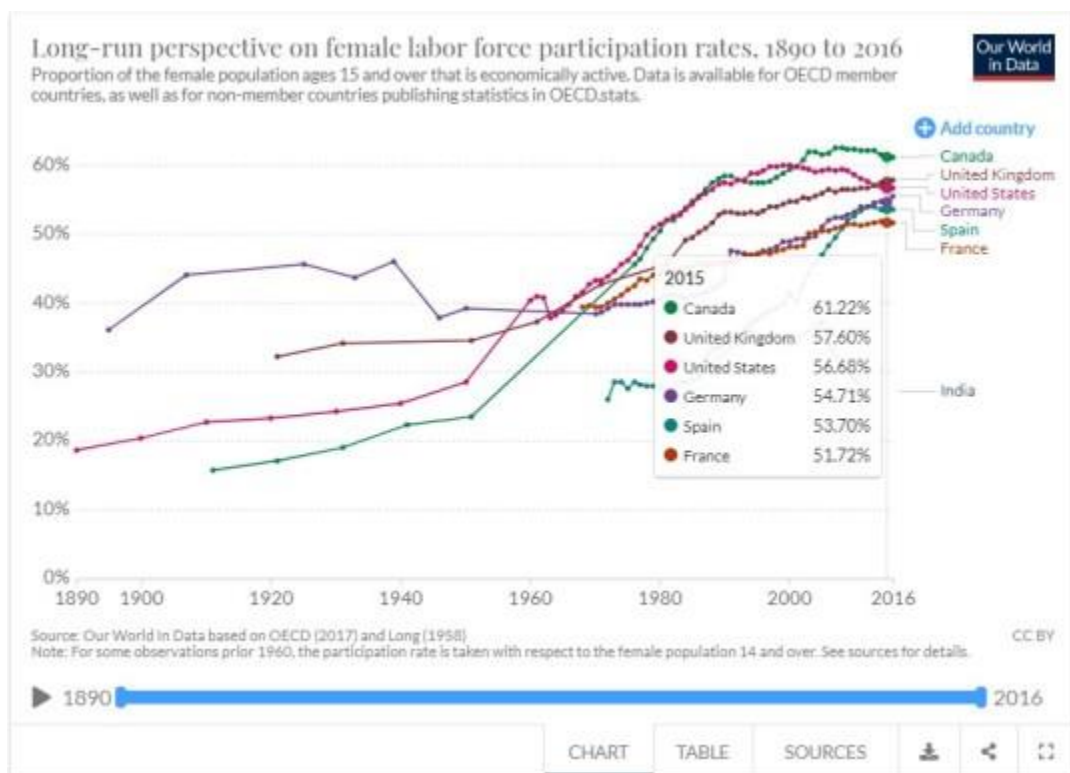
What are the causes of Unemployment?

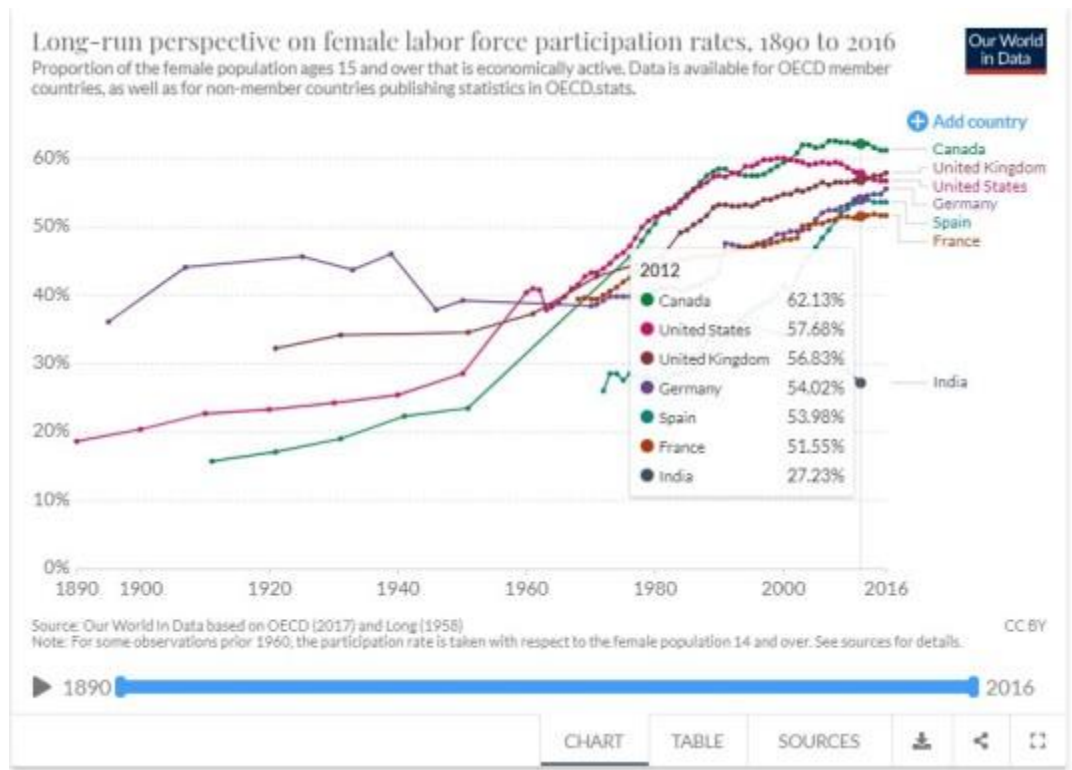
Unemployment is one of the most common and complicated issues the global community has to deal with. Its roots can be found in a number of causes, the most important of which are:

- a country's increased population.
- a country's human resources being uneducated/unqualified.
- the rapid development and spread of technology.
- a great number of low wage jobs/job dissatisfaction.
- Discrimination

Female Employment rate

The 20th century saw a radical increase in the number of women participating in labor markets across early-industrialized countries. The following visualization shows this. It plots long-run female participation rates, piecing together OECD data and available historical estimates for a selection of early-industrialized countries.





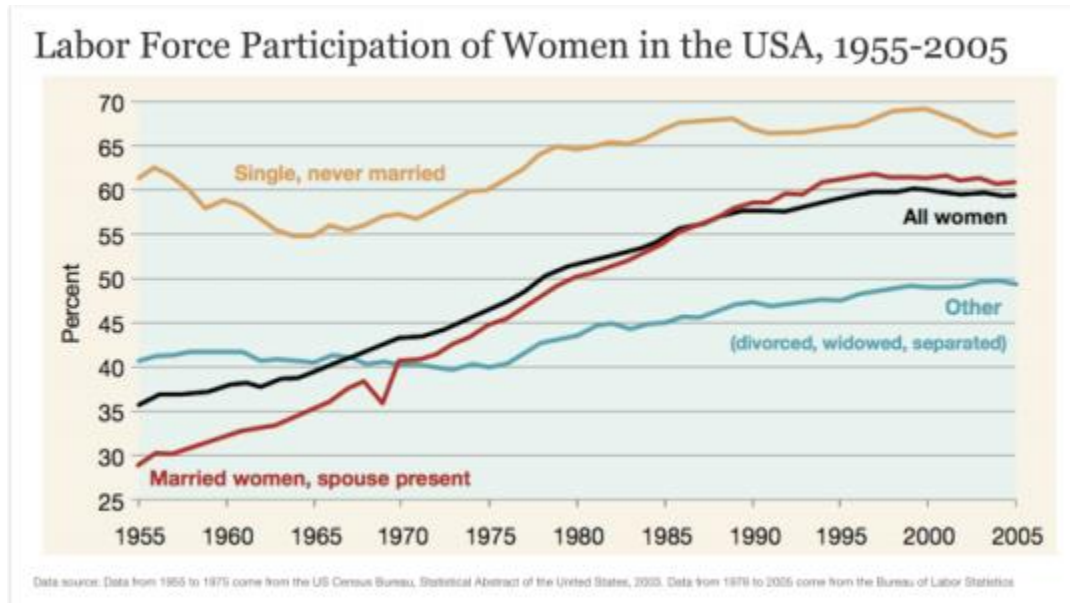
After doing some comparison 2012 and 2015, Canada is leading with highest female employment rate in both the images shown above.

As we can see, there are positive trends across all of these countries. Notably, growth in participation began at different points in time, and proceeded at different rates; nonetheless, the substantial and sustained increases in the labor force participation of women in rich countries remains a striking feature of economic and social change in the 20th century.

Married women drove the increase in female labor force participation in rich countries

Most of the long-run increase in the participation of women in labor markets throughout the last century is attributable specifically to an increase in the participation of married women.

The following visualization shows the experience of the US. It plots female labor force participation rates, differentiating by marital status. As can be seen, the marked upward trend observed for the general female population is mainly driven by the trend among married women. Heckman and Killingsworth (1986) provide evidence of similar historical trends for the UK, Germany and Canada.



Among OECD countries and India, Iceland has highest female employment rate which is 69.59% and India has 32.29% employment rate which is less than half of highest employment rate country.

This is found that Iceland has highest employment rate as well as highest female employment rate.

Start Using Python

I have used 'Gapminder dataset' and choose the topic **employee rate vs female employee rate**. I have taken 2 variables from by topic and subtopic as, **employrate** and **femaleemployrate** and then taken third variable **incomeperperson** from the codebook to show frequency distribution.

First I import my gapminder dataset in "Spyder" and printed complete data of the file.

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 23 20:20:31 2020

@author: smriti
"""

import pandas as pd
import numpy
df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv',low_memory=False)
print(df)
print(len(df))
print(len(df.columns))
print(len(df.index))
```

The output is as follows:

```
In [6]: runcell(0, 'C:/Users/smriti/untitled2.py')
country incomeperperson ... employrate urbanrate
0 Afghanistan ... 55.7000007629394 24.04
1 Albania 1914.99655094922 ... 51.4000015258789 46.72
2 Algeria 2231.99333515006 ... 50.5 65.22
3 Andorra 21943.3398976022 ... 88.92
4 Angola 1381.00426770244 ... 75.6999969482422 56.7
.. ... ...
208 Vietnam 722.807558834445 ... 71 27.84
209 West Bank and Gaza ... 32 71.9
210 Yemen, Rep. 610.3573673206 ... 39 30.64
211 Zambia 432.226336974583 ... 61 35.42
212 Zimbabwe 320.771889948584 ... 66.8000030517578 37.34

[213 rows x 16 columns]
213
16
213
counts of employrate in the past years
```

Then we count the number of employrate and percentage of employrate in past years for which my input commands are:

```
print('counts of employrate in the past years')
c1=df['employrate'].value_counts(sort=False)
print(c1)

print('percentages of employrate in the past years')
p1=df['employrate'].value_counts(sort=False,normalize=True)
print(p1)
```

Output is,

```
counts of employrate in the past years
60.5 1
56.9000015258789 1
65 3
50.9000015258789 2
41.0999984741211 1
..
58.5 1
38.9000015258789 1
80.6999969482422 1
66.5999984741211 1
32 1
Name: employrate, Length: 140, dtype: int64
percentages of employrate in the past years
60.5 0.004695
56.9000015258789 0.004695
65 0.014085
50.9000015258789 0.009390
41.0999984741211 0.004695
...
58.5 0.004695
38.9000015258789 0.004695
80.6999969482422 0.004695
66.5999984741211 0.004695
32 0.004695
Name: employrate, Length: 140, dtype: float64
```

Then we count the number of femaleemployrate and percentage of femaleemployrate in past years for which my input commands are:

```

print('counts of femaleemployrate in the past years')
c2=df['femaleemployrate'].value_counts(sort=False)
print(c2)

print('percentages of femaleemployrate in the past years')
p2=df['femaleemployrate'].value_counts(sort=False,normalize=True)
print(p2)

```

Output is,

```

counts of femaleemployrate in the past years
32.2999992370606    1
35.7999992370606    1
44.0999984741211    1
41.0999984741211    1
45.2999992370606    2
..
23.2000007629395    1
68.9000015258789    2
22.6000003814697    1
36.5                1
66.5999984741211    1
Name: femaleemployrate, Length: 154, dtype: int64
percentages of femaleemployrate in the past years
32.2999992370606    0.004695
35.7999992370606    0.004695
44.0999984741211    0.004695
41.0999984741211    0.004695
45.2999992370606    0.009390
...
23.2000007629395    0.004695
68.9000015258789    0.009390
22.6000003814697    0.004695
36.5                0.004695
66.5999984741211    0.004695
Name: femaleemployrate, Length: 154, dtype: float64
counts of incomeperperson in the past years

```

Then we count the number of incomeperperson and percentage of incomeperperson in past years for which my input commands are:

```

print('counts of incomeperperson in the past years')
c3=df['incomeperperson'].value_counts(sort=False)
print(c3)

print('percentages of incomeperperson in the past yeras')
p3=df['incomeperperson'].value_counts(sort=False,normalize=True)
print(p3)

```

Output is,


```

counts of incomeperperson in the past years
736.268053798916    1
6575.7450438731    1
1728.02097614196    1
115.305995904875    1
558.062876627297    1
..
12729.4543999442    1
9243.58705259742    1
456.385711651118    1
2534.00037997061    1
554.87984007104    1
Name: incomeperperson, Length: 191, dtype: int64
percentages of incomeperperson in the past yeras
736.268053798916    0.004695
6575.7450438731    0.004695
1728.02097614196    0.004695
115.305995904875    0.004695
558.062876627297    0.004695
...
12729.4543999442    0.004695
9243.58705259742    0.004695
456.385711651118    0.004695
2534.00037997061    0.004695
554.87984007104    0.004695
Name: incomeperperson, Length: 191, dtype: float64

```

Input command for the size of employrate is,

```

ctl=df.groupby('employrate').size()
print(ctl)

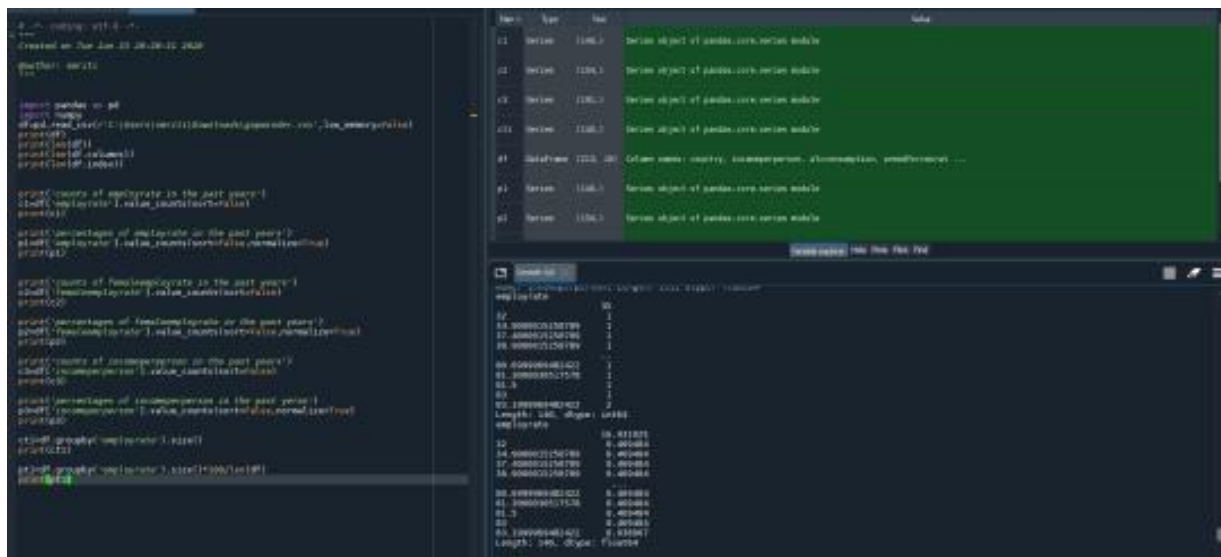
pt1=df.groupby('employrate').size()*100/len(df)
print(pt1)

```

Here, len is the length and pt1 denotes the percentage.

The complete input, variable explorer and output is as follows:

The frequency table of some employrate is as follows:



p1 - Series

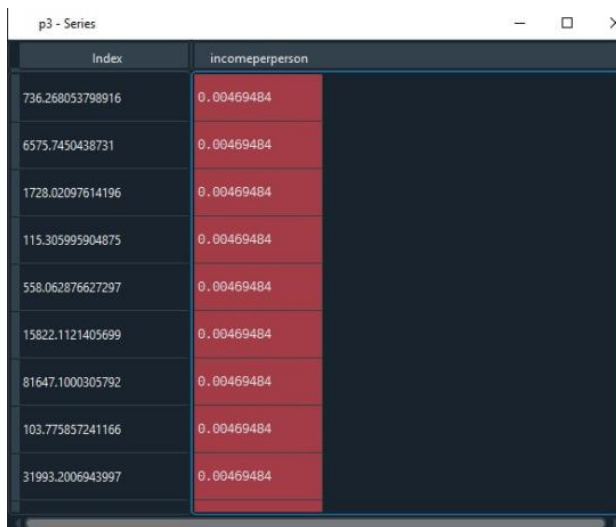
Index	employrate
60.5	0.00469484
56.9000015258789	0.00469484
65	0.0140845
50.9000015258789	0.00938967
41.0999984741211	0.00469484
70.4000015258789	0.00938967
51.4000015258789	0.00469484
55.0999984741211	0.00469484
62.2999992370606	0.00469484

The frequency table of some femaleemployrate is as follows:

p2 - Series

Index	femaleemployrate
32.2999992370606	0.00469484
35.7999992370606	0.00469484
44.0999984741211	0.00469484
41.0999984741211	0.00469484
45.2999992370606	0.00938967
13	0.00469484
69	0.00469484
46.2000007629394	0.00469484
65	0.00469484

The frequency table of some incomeperperson is as follows:



The screenshot shows a Jupyter Notebook window titled 'p3 - Series'. It displays a frequency table for the variable 'incomeperperson'. The table has two columns: 'Index' and 'incomeperperson'. The 'Index' column contains numerical values, and the 'incomeperperson' column contains the frequency of each index. The table is displayed in a dark-themed interface with red highlighting for the frequency values.

Index	incomeperperson
736.268053798916	0.00469484
6575.7450438731	0.00469484
1728.02097614196	0.00469484
115.305995904875	0.00469484
558.062876627297	0.00469484
15822.1121405699	0.00469484
81647.1000305792	0.00469484
103.775857241166	0.00469484
31993.2006943997	0.00469484

The employee rate is more in all countries as compare to female employee rate.

Making Data Management Decisions

Now, I am in the phase of making Data management decisions, here is the complete library of my program that is,

```
import pandas as pd
import numpy as np
from numpy import nan
import statsmodels.formula.api as smf
import statsmodels.stats.multicomp as multi
import scipy
from scipy.stats import pearsonr
```

Now, i have imported my dataset and i showed here data of first 20 rows that is,

```
df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv')
print(df)
print(len(df))
print(len(df.columns))
print(len(df.index))
print(df.head(20))
```

The output is,

```
country  incomeperperson  ...  employrate  urbanrate
0  Afghanistan  ...  55.7000007629394  24.04
1  Albania  ...  51.4000015258789  46.72
2  Algeria  ...  50.5  65.22
3  Andorra  ...  88.92
4  Angola  ...  75.6999969482422  56.7
5  Antigua and Barbuda  ...  30.46
6  Argentina  ...  58.4000015258789  92
7  Armenia  ...  40.0999984741211  63.86
8  Aruba  ...  46.78
9  Australia  ...  61.5  88.74
10  Austria  ...  57.0999984741211  67.16
11  Azerbaijan  ...  60.9000015258789  51.92
12  Bahamas  ...  66.5999984741211  83.7
13  Bahrain  ...  60.4000015258789  88.52
14  Bangladesh  ...  68.0999984741211  27.14
15  Barbados  ...  66.9000015258789  39.84
16  Belarus  ...  53.4000015258789  73.46
17  Belgium  ...  48.5999984741211  97.36
18  Belize  ...  56.7999992370606  51.7
19  Benin  ...  71.5999984741211  41.2

[20 rows x 16 columns]
```

Now, this is the complete data in 20 rows, i have chosen three variables that is, **employrate**, **femaleemployrate** and **incomeperperson**.

Now, the data for **employrate** is,

```
print(df['employrate'].head(20))

0    55.7000007629394
1    51.4000015258789
2         50.5
3
4    75.6999969482422
5
6    58.4000015258789
7    40.0999984741211
8
9         61.5
10   57.0999984741211
11   60.9000015258789
12   66.5999984741211
13   60.4000015258789
14   68.0999984741211
15   66.9000015258789
16   53.4000015258789
17   48.5999984741211
18   56.7999992370606
19   71.5999984741211
Name: employrate, dtype: object
```

Here you can see some missing values such as in row 3,5 and 8.

The data for **femaleemployrate** is,

```
print(df['femaleemployrate'].head(20))
```

```
0    25.6000003814697
1    42.0999984741211
2    31.7000007629394
3
4    69.4000015258789
5
6    45.9000015258789
7    34.2000007629394
8
9    54.5999984741211
10   49.7000007629394
11   56.2000007629394
12   60.7000007629394
13   30.2000007629394
14   53.5999984741211
15   60.2999992370606
16   48.5999984741211
17   41.7000007629394
18   38.7999992370606
19   58.2000007629394
Name: femaleemployrate, dtype: object
```

Here you can see some missing values such as in row 3,5 and 8.

The data for **incomeperperson** is,

```
print(df['incomeperperson'].head(20))
```

```
0
1    1914.99655094922
2    2231.99333515006
3    21943.3398976022
4    1381.00426770244
5    11894.4640745081
6    10749.4192379463
7    1326.74175718861
8
9    25249.98606148
10   26692.9841066319
11   2344.89691619809
12   19630.5405471267
13   12505.2125447354
14   558.062876627297
15   9243.58705259742
16   2737.67037938365
17   24496.0482640925
18    3545.652173906
19   377.039699461343
```

Here you can see some missing values such as in row 8.

1. Code out missing values

Coding out missing values for employrate:

The input is,

```
print(df['employrate'].head(20))
df['employrate']=df['employrate'].head(20).replace(' ',numpy.nan)

a1=df['employrate'].value_counts(sort=False,dropna=False)
print(a1)
```

The output is,

```
NaN      196
75.6999969482422      1
58.4000015258789      1
66.9000015258789      1
57.0999984741211      1
68.0999984741211      1
71.5999984741211      1
60.4000015258789      1
48.5999984741211      1
50.5      1
56.7999992370606      1
61.5      1
66.5999984741211      1
51.4000015258789      1
55.7000007629394      1
40.0999984741211      1
53.4000015258789      1
60.9000015258789      1
Name: employrate, dtype: int64
```

Here, you can see the missing values in employrate are out.

Similarly, for other two variables,

Coding out missing values for femaleemployrate:

The input is,

```
print(df['femaleemployrate'].head(20))
df['femaleemployrate']=df['femaleemployrate'].head(20).replace(' ',numpy.nan)

a2=df['femaleemployrate'].value_counts(sort=False,dropna=False)
print(a2)
```


The output is,

```
NaN      196
25.6000003814697    1
41.7000007629394    1
60.7000007629394    1
54.5999984741211    1
42.0999984741211    1
69.4000015258789    1
58.2000007629394    1
60.2999992370606    1
49.7000007629394    1
45.9000015258789    1
38.7999992370606    1
30.2000007629394    1
56.2000007629394    1
31.7000007629394    1
53.5999984741211    1
48.5999984741211    1
34.2000007629394    1
Name: femaleemployrate, dtype: int64
```

Coding out missing values for incomeperperson:

The input is,

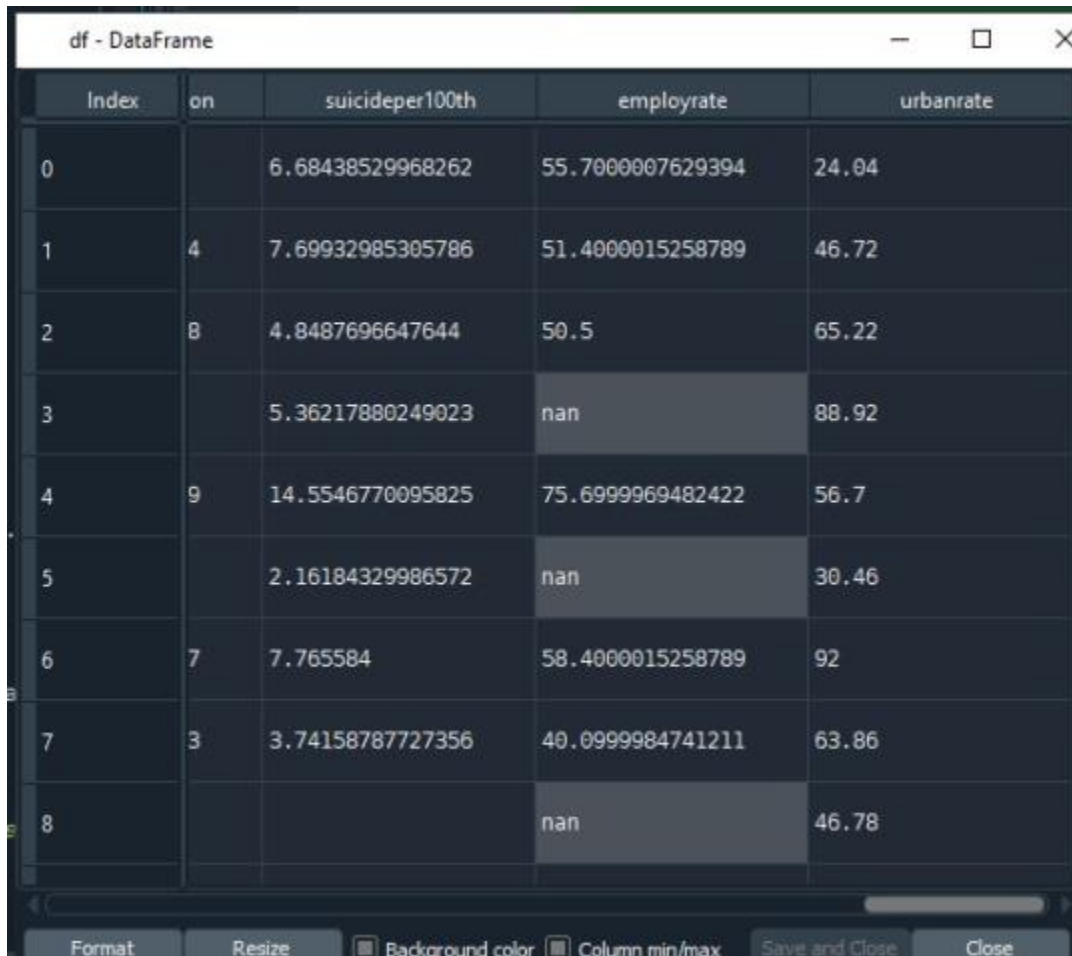
```
print(df['incomeperperson'].head(20))
df['incomeperperson']=df['incomeperperson'].head(20).replace(' ',numpy.nan)

a3=df['incomeperperson'].value_counts(sort=False,dropna=False)
print(a3)
```

The output is,

```
Name: incomeperperson, dtype: object
NaN      195
25249.98606148    1
377.039699461343    1
26692.9841066319    1
3545.652173906    1
9243.58705259742    1
10749.4192379463    1
21943.3398976022    1
2344.89691619809    1
1914.99655094922    1
1381.00426770244    1
11894.4640745081    1
24496.0482640925    1
1326.74175718861    1
2737.67037938365    1
2231.99333515006    1
558.062876627297    1
12505.2125447354    1
19630.5405471267    1
Name: incomeperperson, dtype: int64
```


You can also see variable explorer of employrate where missing values are replaced with nan,



df - DataFrame

Index	on	suicideper100th	employrate	urbanrate
0		6.68438529968262	55.7000007629394	24.04
1	4	7.69932985305786	51.4000015258789	46.72
2	8	4.8487696647644	50.5	65.22
3		5.36217880249023	nan	88.92
4	9	14.5546770095825	75.6999969482422	56.7
5		2.16184329986572	nan	30.46
6	7	7.765584	58.4000015258789	92
7	3	3.74158787727356	40.0999984741211	63.86
8			nan	46.78

Format Resize ☐ Background color ☐ Column min/max Save and Close Close

2. Check in how many countries, few columns are missing:

Checking for lifeexpectancy, breastcancerper100th, suicideper100th and my selected variables, **employrate**, **femaleemployrate** and **incomeperperson**:

```
def check_missing_values(df,cols):
    for col in cols:
        print("Column {} is missing:".format(col))
        print((df[col].values==' ').sum())
        print()

def convert_numeric(df1,cols):
    for col in cols:
        df1[col]=pd.to_numeric(df1[col],errors='coerce')

df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv')
print("Null values:")
print(df.isnull().values.any())

cols=['lifeexpectancy','breastcancerper100th','suicideper100th']
norm_cols=['employrate','femaleemployrate','incomeperperson']

df2=df.copy()

check_missing_values(df, cols)
check_missing_values(df, norm_cols)

convert_numeric(df2, cols)
convert_numeric(df2, norm_cols)
```

The output is,

```
Null values:
False
Column lifeexpectancy is missing:
22
Column breastcancerper100th is missing:
40
Column suicideper100th is missing:
22
Column employrate is missing:
35
Column femaleemployrate is missing:
35
Column incomeperperson is missing:
23
```

Here, the **column employrate** is **missing in 35 countries**,
The **column femaleemployrate** is **missing in 35 countries** and
The **column incomeperperson** is **missing in 23 countries**.

Therefore, by this data, employrate and femaleemployrate is missing equally and incomeperperson is somehow less missed.

2. Analysis of variance:

We can transform the data from continuous to quantitative by selecting a category and binning the variable in question, dividing it into percentiles. The independent variable will be converted into a categorical variable, while the dependent variable will stay continuous. We can use the `qcut()` function in Pandas to divide the dataframe into bins

```
def bin(df1,cols):
    for col in cols:
        new_col_name="{0}_bins".format(col)
        df1[new_col_name]=pd.qcut(df1[col],10,labels=['1=10%','2=20%','3=30%','4=40%','5=50%','6=60%','7=70%','8=80%','9=90%','10=100%'])
df3=df2.copy()
# This creates new columns filled with the binned column data
bin(df3,cols)
bin(df3,norm_cols)
```

After the variables have been transformed and are ready to be analyzed, we can use the statsmodel library to carry out an ANOVA on the selected features. We'll print out the results of the ANOVA and check to see if the relationship between the two variables is statistically significant:

```
def bin(df1,cols):
    for col in cols:
        new_col_name="{0}_bins".format(col)
        df1[new_col_name]=pd.qcut(df1[col],10,labels=['1=10%','2=20%','3=30%','4=40%','5=50%','6=60%','7=70%','8=80%','9=90%','10=100%'])
df3=df2.copy()
bin(df3,cols)
bin(df3,norm_cols)

anova_df = df3[['incomeperperson', 'femaleemployrate_bins', 'employrate_bins']].dropna()
relate_df = df3[['incomeperperson', 'femaleemployrate_bins']]
anova = smf.ols(formula='incomeperperson ~ C(femaleemployrate_bins)', data=anova_df).fit()
print(anova.summary())

# We may also want to check the mean and standard deviation for the groups
mean = relate_df.groupby('femaleemployrate_bins').mean()
sd = relate_df.groupby('femaleemployrate_bins').std()
print(mean)
print(sd)
```

The output of the model is,

```

=====
                   OLS Regression Results
=====
Dep. Variable:      incomeperperson      R-squared:      0.136
Model:              OLS                  Adj. R-squared:  0.087
Method:             Least Squares        F-statistic:     2.737
Date:               Fri, 26 Jun 2020      Prob (F-statistic): 0.00541
Time:               11:58:36              Log-Likelihood: -1764.8
No. Observations:   166                  AIC:            3550.
Df Residuals:       156                  BIC:            3581.
Df Model:           9
Covariance Type:    nonrobust
=====
                    coef    std err          t      P>|t|      [0.025      0.975]
-----
Intercept          3660.2676    2583.548      1.417      0.159    -1442.982    8763.518
C(femaleemployrate_bins)[T.2=20%]  1261.5345    3550.747      0.355      0.723    -5752.212    8275.281
C(femaleemployrate_bins)[T.3=30%]  3960.5156    3714.084      1.066      0.288    -3375.869    1.13e+04
C(femaleemployrate_bins)[T.4=40%]  1867.7604    3714.084      0.503      0.616    -5468.624    9204.144
C(femaleemployrate_bins)[T.5=50%]  9389.6390    3653.689      2.570      0.011    2172.554    1.66e+04
C(femaleemployrate_bins)[T.6=60%]  2072.5226    3550.747      0.584      0.560    -4941.224    9086.269
C(femaleemployrate_bins)[T.7=70%]  7970.1875    3653.689      2.181      0.031     753.022    1.52e+04
C(femaleemployrate_bins)[T.8=80%]  1.125e+04    3599.557      3.125      0.002    4137.706    1.84e+04
C(femaleemployrate_bins)[T.9=90%]  3553.8123    3599.557      0.987      0.325    -3556.347    1.07e+04
C(femaleemployrate_bins)[T.10=100%] -1226.2167    3550.747     -0.345      0.730    -8239.963    5787.530
=====

Omnibus:           45.371    Durbin-Watson:      1.818
Prob(Omnibus):     0.000    Jarque-Bera (JB):    77.492
Skew:              1.372    Prob(JB):            1.49e-17
Kurtosis:          4.918    Cond. No.             11.1
=====

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
incomeperperson
```

We can see that the model gives a very small P-value (Prob F-statistic) of 0.00541 . This is far less than the usual significance threshold of 0.05, so we conclude there is a significant relationship between income per person and female employee rate.

3. Recoding variables:

We'll make a few different comparisons using a recoding scheme and map the records into new feature columns.

The input for the model is,

```
def half_bin(df1, cols):
    for col in cols:
        new_col_name = "{}_bins_2".format(col)
        df1[new_col_name] = pd.qcut(df1[col], 2, labels=["1=50%", "2=100%"])

half_bin(df3, ['femaleemployrate'])

# Recoding scheme
recode_2 = {"3=30%": "3=30%", "7=70%": "7=70%"}
recode_3 = {"2=20%": "2=20%", "8=80%": "8=80%"}
recode_4 = {"6=60%": "6=60%", "9=90%": "9=90%"}
recode_5 = {"4=40%": "4=40%", "7=70%": "7=70%"}

# Create the new features
df3['Comp_3v7'] = df3['incomeperperson_bins'].map(recode_2)
df3['Comp_2v8'] = df3['incomeperperson_bins'].map(recode_3)
df3['Comp_6v9'] = df3['incomeperperson_bins'].map(recode_4)
df3['Comp_4v7'] = df3['incomeperperson_bins'].map(recode_5)

count_table = pd.crosstab(df3['femaleemployrate_bins_2'], df3['incomeperperson_bins'])
print(count_table.head(10))
```

The output is,

```
femaleemployrate_bins    incomeperperson
1=10%                    3660.267597
2=20%                    4921.802120
3=30%                    7620.783174
4=40%                    5528.027964
5=50%                    13049.906615
6=60%                    5732.790223
7=70%                    11630.375108
8=80%                    14908.132995
9=90%                    7214.079848
10=100%                  2434.050869
incomeperperson
femaleemployrate_bins
1=10%                    3399.507431
2=20%                    5749.979711
3=30%                    9107.811927
4=40%                    6351.651046
5=50%                    15375.571447
6=60%                    7348.898137
7=70%                    13557.557168
8=80%                    15483.860973
9=90%                    11415.036386
10=100%                  7882.834624
incomeperperson_bins
femaleemployrate_bins_2  1=10%  2=20%  3=30%  ...  8=80%  9=90%  10=100%
1=50%                   2       3       9  ...    11     8       3
2=100%                  17      16       8  ...     5     9      12

[2 rows x 10 columns]
```

4. Use of pearsonr() function from scipy to carry out correlation:

The input is,

```
df_clean = df2.dropna()
df_clean['incomeperperson'] = df_clean['incomeperperson'].replace('', np.nan)

print('Assoc. - female employee rate and income per person')
print(pearsonr(df_clean['femaleemployrate'], df_clean['incomeperperson']))

print('Assoc. - between employment rate and income per person')
print(pearsonr(df_clean['employrate'], df_clean['incomeperperson']))
```

The output of the model is,

```
Assoc. - female employee rate and income per person
(0.021121311194267543, 0.7902847605862906)
Assoc. - between employment rate and income per person
(-0.008526427248756399, 0.9145135443378481)
```

The first value is the direction and strength of the correlation, while the second is the P-value. The numbers suggest a **fairly strong correlation** between female employee rate and income per person that isn't due to chance. Meanwhile, there's a **weaker, though still significant correlation** between employment rate and income per person.

5. Choosing suitable variable as secondary variable:

Let's try income level per person and divide it into three different groups:

```
def income_groups(row):
    if row['incomeperperson'] <= 744.23:
        return 1
    elif row['incomeperperson'] <= 942.32:
        return 2
    else:
        return 3

# Apply function and set the new features in the dataframe
df_clean['income_group'] = df_clean.apply(lambda row: income_groups(row), axis=1)

# Create a few subframes to try test for moderation
subframe_1 = df_clean[(df_clean['income_group'] == 1)]
subframe_2 = df_clean[(df_clean['income_group'] == 2)]
subframe_3 = df_clean[(df_clean['income_group'] == 3)]

print('Assoc. -employee rate and female employee rate for low income countries')
print(pearsonr(subframe_1['employrate'], subframe_1['femaleemployrate']))

print('Assoc. - employee rate and female employee rate for medium income countries')
print(pearsonr(subframe_2['employrate'], subframe_2['femaleemployrate']))

print('Assoc. - employee rate and female employee rate for high income countries')
print(pearsonr(subframe_3['employrate'], subframe_3['femaleemployrate']))
```


The output is,

```
Assoc. -employee rate and female employee rate for low income countries  
(0.9295021527952501, 1.1160563249533423e-20)  
Assoc. - employee rate and female employee rate for medium income countries  
(0.7628727173433518, 0.44757656638718524)  
Assoc. - employee rate and female employee rate for high income countries  
(0.7664357771213115, 7.011767293090408e-23)
```

Once more, the first value is the direction and strength of the correlation, while the second is the P-value.

Conclusion:

statsmodels is an extremely useful library that allows Python users to analyze data and run statistical tests on datasets. You can carry out ANOVAs, Chi-Square Tests, Pearson Correlations and test for moderation.

GRAPHICAL ANALYSIS

UNIVARIATE GRAPHICAL ANALYSIS

After analyzing the variables numerically, we will shift our domain to graphical approach.

I have chosen three variables that is, **incomeperperson**, **employrate** and **femaleemployrate**

1.

Income per person:

We will investigate this data by 2 ways, Categorical Count Plot and Distributive Plot.

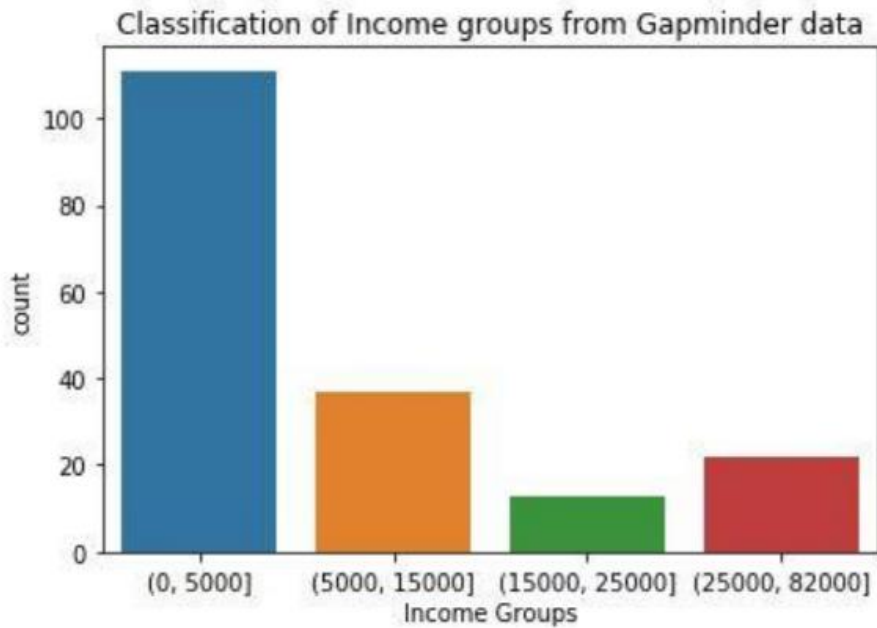
Categorical Count Plot:

To plot incomes by converting into categories, we will first cut the range of incomes into four intervals. The first interval is from 0 to 5000, second is from 5000 to 15000, third is from 15000 to 25000 and last is from 25000 to 82000.

The frequency table obtained is shown below:

```
(0, 5000]      111  
(5000, 15000]  37  
(15000, 25000] 13  
(25000, 82000] 22  
Name: incomegroups, dtype: int64
```

And the graph obtained for these counts is:



This proves that our numeric interpretation of this variable was correct. Out of 183 countries, 111 have per person income between 0 and 5000. Also, just 22 incomes lie between a great range of 25000 and 82000.

Distributive Plot:

Now we will plot the same set of values by using the “distplot” command in python.

The following code is added to our program:

```
import pandas as pd
import numpy as np
from numpy import nan
import seaborn as sns
import matplotlib.pyplot as plt

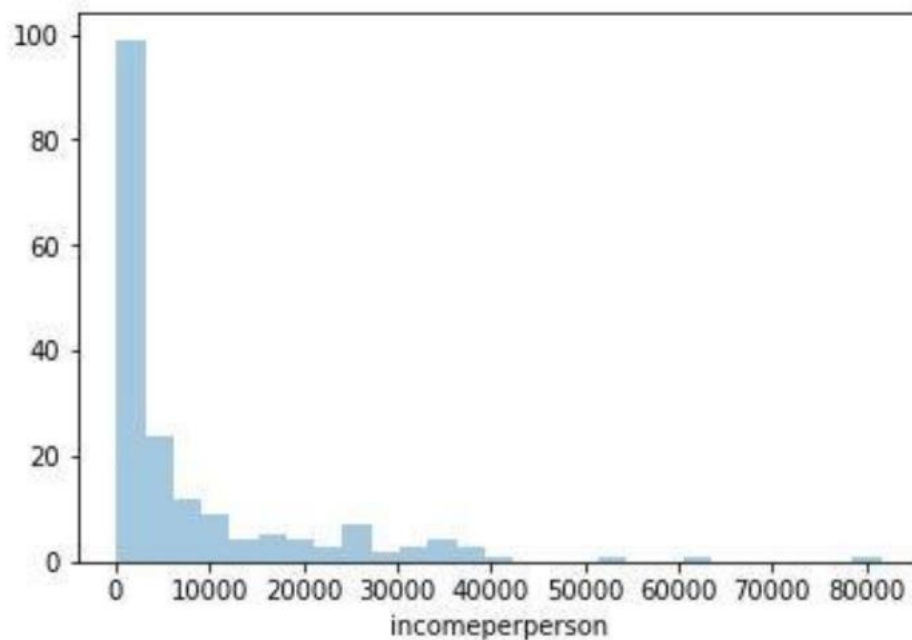
df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv',low_memory=False)

df['incomeperperson']=df['incomeperperson'].apply(pd.to_numeric, errors='coerce')
df['employrate']=df['employrate'].apply(pd.to_numeric, errors='coerce')
df['femaleemployrate']=df['femaleemployrate'].apply(pd.to_numeric, errors='coerce')

sub1=df[(df['incomeperperson']>0)&(df['employrate']>0)&(df['femaleemployrate']>0)]
sub2=sub1.copy()

sns.distplot(sub2['incomeperperson'].dropna(),kde=False)
plt.title('Income per person in different countries')
```

The graph obtained against this code is shown below:



This automatically generated graph over equally distributed range also depicts loudly that more countries exist with lesser per person income. The bars are higher in the starting phase and start decreasing gradually as we move right. The bars disappear after 40000, showing that not much countries are there with average income more than 40000.

2. Employment Rate:

Now we will plot the same set of values by using the “distplot” command in python.

The following code is added to our program:

```
import pandas as pd
import numpy as np
from numpy import nan
import seaborn as sns
import matplotlib.pyplot as plt

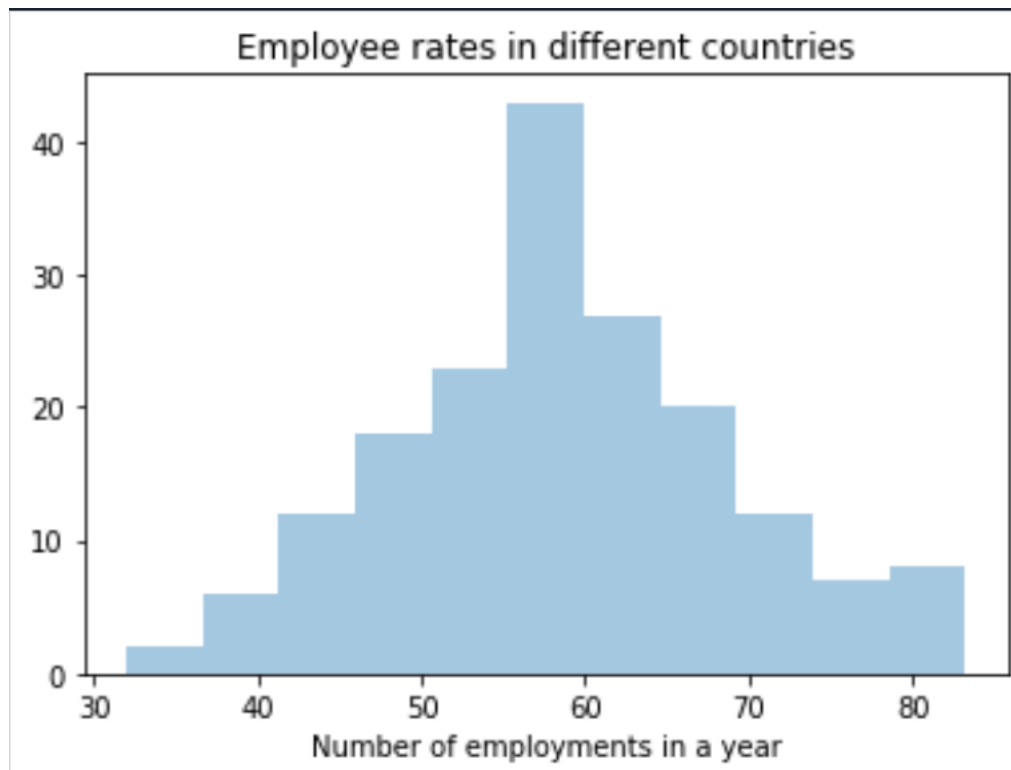
df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv',low_memory=False)

df['incomeperperson']=df['incomeperperson'].apply(pd.to_numeric, errors='coerce')
df['employrate']=df['employrate'].apply(pd.to_numeric, errors='coerce')
df['femaleemployrate']=df['femaleemployrate'].apply(pd.to_numeric, errors='coerce')

sub1=df[(df['incomeperperson']>0)&(df['employrate']>0)&(df['femaleemployrate']>0)]
sub2=sub1.copy()

sns.distplot(sub2['employrate'].dropna(),kde=False)
plt.title('Employment rates in different countries')
```

The graph obtained against this code is shown below:



Description of Employment rate:

We make use of `.describe()` method to show the result in more clear way:

The input for this is,

```
print('Describe the data for employment rates')
desc1=sub2['employrate'].describe()
print(desc1)
```

The output is,

```
Describe the data for employment rates
count    166.000000
mean      58.987952
std       10.327332
min       34.900002
25%       51.575001
50%       58.850000
75%       64.975000
max       83.199997
Name: employrate, dtype: float64
```

The total number of countries with employment rate is 166.

Here, it is clearly shown that **on an average, 58% employment rate is there in different countries**. The standard deviation is about 11%, suggesting that there is quite a bit of variability from country to country in terms of the employment rate living in different countries.

3. Female Employment Rate:

Now we will plot the same set of values by using the “distplot” command in python.

The following code is added to our program:

```
import pandas as pd
import numpy as np
from numpy import nan
import seaborn as sns
import matplotlib.pyplot as plt

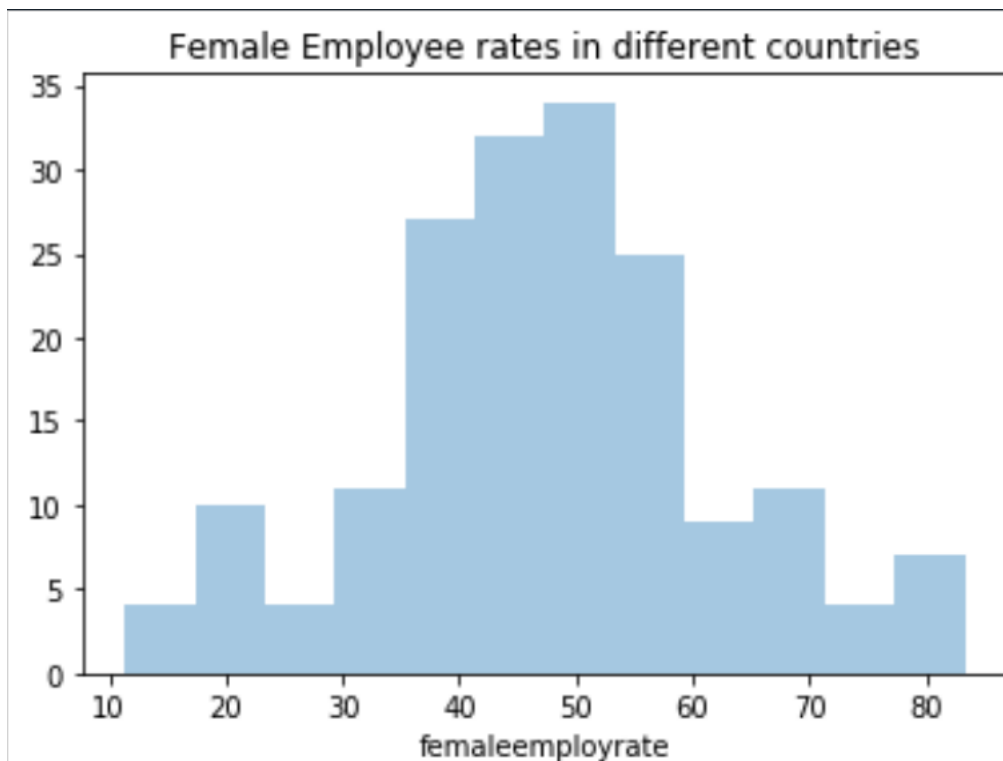
df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv',low_memory=False)

df['incomeperperson']=df['incomeperperson'].apply(pd.to_numeric, errors='coerce')
df['employrate']=df['employrate'].apply(pd.to_numeric, errors='coerce')
df['femaleemployrate']=df['femaleemployrate'].apply(pd.to_numeric, errors='coerce')

sub1=df[(df['incomeperperson']>0)&(df['employrate']>0)&(df['femaleemployrate']>0)]
sub2=sub1.copy()

sns.distplot(sub2['femaleemployrate'].dropna(),kde=False)
plt.title('Female Employment rates in different countries')
```

The graph obtained against this code is shown below:



Description of Female Employment rate:

We make use of **.describe()** method to show the result in more clear way:

The input for this is,

```
print('Describe the data for Female employment rates')
desc2=sub2['femaleemployrate'].describe()
print(desc2)
```

The output is,

```
Describe the data for Female employment rates
count    166.000000
mean      47.963855
std       14.618138
min       12.400000
25%       39.250001
50%       48.450001
75%       56.150001
max       83.300003
Name: femaleemployrate, dtype: float64
```

The total number of countries with female employment rate is 166.

Here, it is clearly shown that **on an average, 48% female employment rate is there in different countries**. The standard deviation is about 15%, suggesting that there is quite a bit of variability from country to country in terms of the female employment rate living in different countries.

Conclusion:

Comparing employrate and femaleemployrate, on an average employment rate is more than female employment rate among different countries.

Bivariate Graphical Analysis

Now we move to the final step of our research, the bivariate analysis of our variables. From the previous observations and explanations, we are expecting to see a relationship between all the variables (employment rate and female employment rate), (income per person and employment rate) and (income per person and female employment rate).

1. Bivariate graphical analysis for employment rate and female employment rate:

To obtain the final plot between both these variables by using “seaborn.regplot”, add the following code to the program:

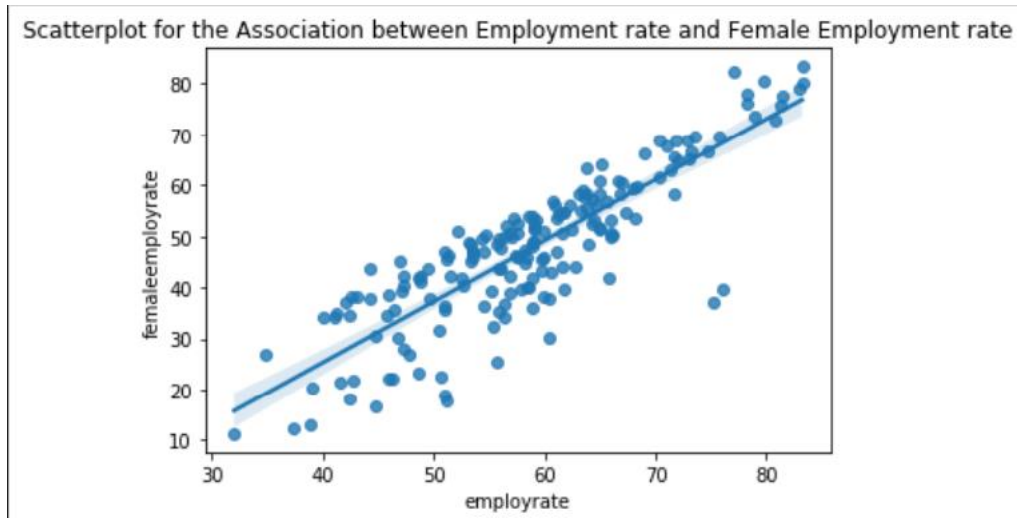
```
import pandas as pd
import numpy as np
from numpy import nan
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv',low_memory=False)

df['incomeperperson']=df['incomeperperson'].apply(pd.to_numeric, errors='coerce')
df['employrate']=df['employrate'].apply(pd.to_numeric, errors='coerce')
df['femaleemployrate']=df['femaleemployrate'].apply(pd.to_numeric, errors='coerce')

sns.regplot(x='employrate',y='femaleemployrate',fit_reg=True,data=df)
plt.title("Scatterplot for the Association between Employment rate and Female Employment rate")
plt.show()
```

The graph obtained by the above code is:



If we set `fit_reg=False` then this line will remove which is shown below:

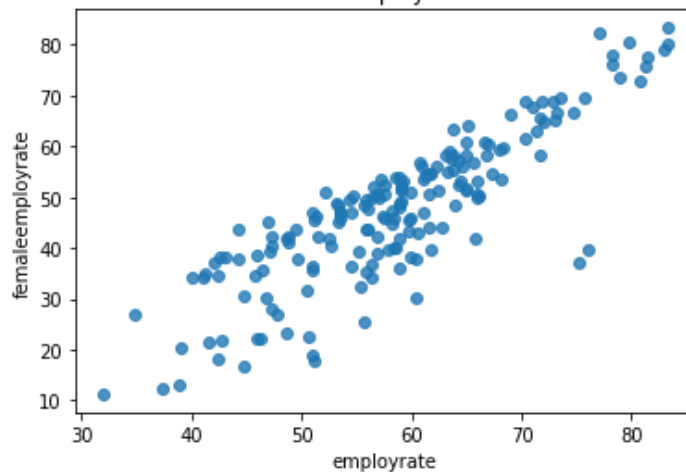
```
import pandas as pd
import numpy as np
from numpy import nan
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv',low_memory=False)

df['incomeperperson']=df['incomeperperson'].apply(pd.to_numeric, errors='coerce')
df['employrate']=df['employrate'].apply(pd.to_numeric, errors='coerce')
df['femaleemployrate']=df['femaleemployrate'].apply(pd.to_numeric, errors='coerce')

sns.regplot(x='employrate',y='femaleemployrate',fit_reg=False,data=df)
plt.title("Scatterplot for the Association between Employment rate and Female Employment rate")
plt.show()
```

Scatterplot for the Association between Employment rate and Female Employment rate



In this scatter plot, the data points follow the linear pattern quite closely. This is an example of a very strong relationship.

Therefore, there is strong relationship between employrate and femaleemployrate.

2. Bivariate graphical analysis for income per person and employment rate:

To obtain the final plot between both these variables by using “seaborn.regplot”, add the following code to the program:

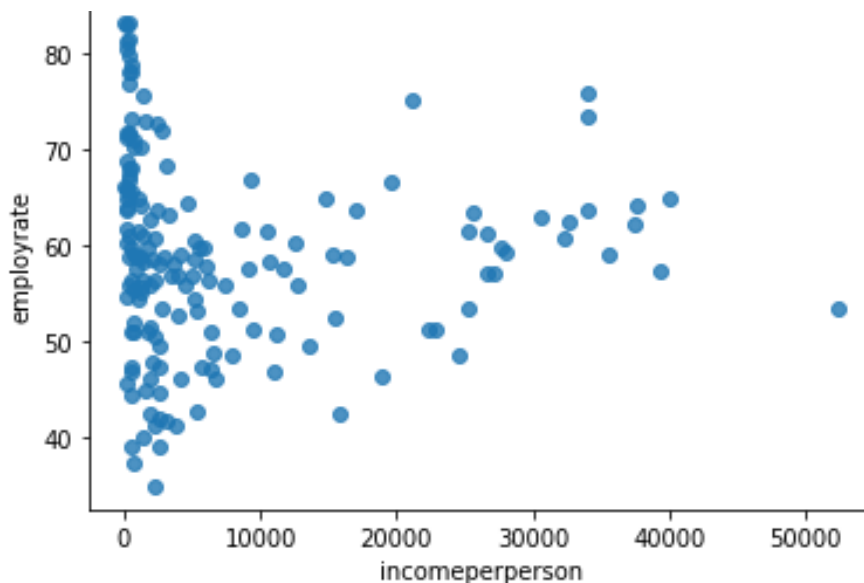
```
import pandas as pd
import numpy as np
from numpy import nan
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv',low_memory=False)

df['incomeperperson']=df['incomeperperson'].apply(pd.to_numeric, errors='coerce')
df['employrate']=df['employrate'].apply(pd.to_numeric, errors='coerce')
df['femaleemployrate']=df['femaleemployrate'].apply(pd.to_numeric, errors='coerce')

sns1=sns.regplot(x='incomeperperson',y='employrate',fit_reg=False,data=df)
plt.title("Scatterplot for the Association between Income per person and Employment rate")
plt.show()
```

The output is,



In this scatter plot, the points also follow the linear pattern, and predicts that the income per person lies more in between 10000 and 20000 for employment rate, few people have income in between range of 20000 and 30000, while less people have income in the range of 30000 and 40000, only 1 person has income range around 50,000 among different countries.

3. Bivariate graphical analysis for income per person and female employment rate:

To obtain the final plot between both these variables by using “seaborn.regplot”, add the following code to the program:

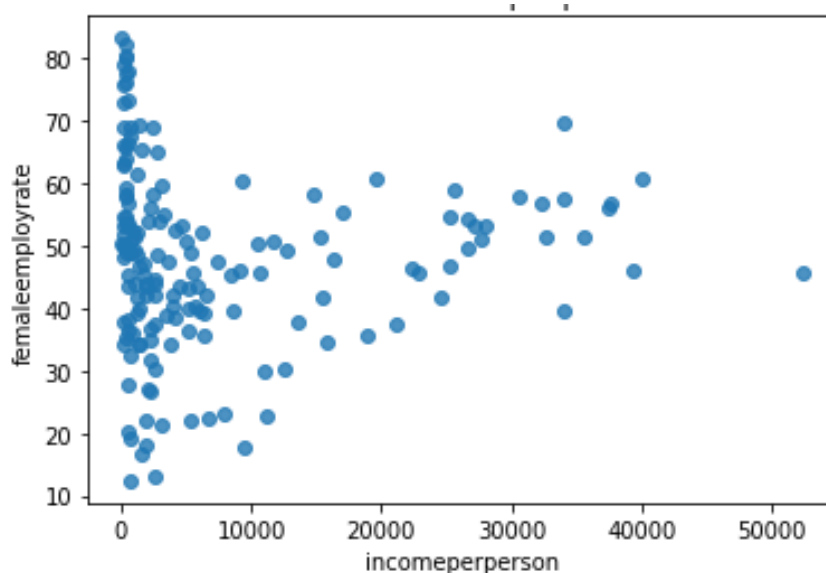
```
import pandas as pd
import numpy as np
from numpy import nan
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv(r'C:\Users\smriti\Downloads\gapminder.csv',low_memory=False)

df['incomeperperson']=df['incomeperperson'].apply(pd.to_numeric, errors='coerce')
df['employrate']=df['employrate'].apply(pd.to_numeric, errors='coerce')
df['femaleemployrate']=df['femaleemployrate'].apply(pd.to_numeric, errors='coerce')

sns2=sns.regplot(x='incomeperperson',y='femaleemployrate',fit_reg=False,data=df)
plt.title("Scatterplot for the Association between Income per person and Female Employment rate")
plt.show()
```

The output is,



In this scatter plot, the points also follow the linear pattern, and predicts that the income per person lies more in between 10000 and 20000 for female employment rate, few people have income in between range of 20000 and 30000, while less people have income in the range of 30000 and 40000, only 1 person has income range around 50,000 among different countries.

Conclusion:

The total number of countries with employment rate and female employment rate is 166 that is, number of countries having employment rate and female employment rate are equal in counts but the countries may be different.

Comparing employrate and femaleemployrate, on an average employment rate is more than female employment rate among different countries.

The standard deviation for employment rate is about 11% and for female employment rate is 15%, suggesting that there is quite a bit of variability from country to country in terms of the employment rate and female employment rate living in different countries.

Now, when we look at the scatter plot of employment rate and female employment rate, the data points follow the linear pattern quite closely which shows that they have very strong relationship.

Therefore, we conclude that there is strong relationship between employrate and femaleemployrate.

For employment rate and female employment rate, it is predicted that the income per person lies more in between 10000 and 20000 for employment rate and female employment rate, few people have income in between range of 20000 and 30000, while less people have income in the range of 30000 and 40000, only 1 person has income range around 50,000 among different countries.