Data Analysis with Python: Zero to Pandas - Course Project Guidelines

(remove this cell before submission)

Important links:

- Make submissions here: https://jovian.ml/learn/data-analysis-with-python-zero-to-pandas/assignment/course-project
- Ask questions here: https://jovian.ml/forum/t/course-project-on-exploratory-data-analysis-discuss-and-share-your-work/11684
- Find interesting datasets here: https://jovian.ml/forum/t/recommended-datasets-for-course-project/11711

This is the starter notebook for the course project for Data Analysis with Python: Zero to Pandas. You will pick a real-world dataset of your choice and apply the concepts learned in this course to perform exploratory data analysis. Use this starter notebook as an outline for your project . Focus on documentation and presentation - this Jupyter notebook will also serve as a project report, so make sure to include detailed explanations wherever possible using Markdown cells.

Evaluation Criteria

Your submission will be evaluated using the following criteria:

- Dataset must contain at least 3 columns and 150 rows of data
- You must ask and answer at least 4 questions about the dataset
- Your submission must include at least 4 visualizations (graphs)
- Your submission must include explanations using markdown cells, apart from the code.
- Your work must not be plagiarized i.e. copy-pasted for somewhere else.

Follow this step-by-step guide to work on your project.

Step 1: Select a real-world dataset

- Find an interesting dataset on this page: https://www.kaggle.com/datasets?fileType=csv
- The data should be in CSV format, and should contain at least 3 columns and 150 rows
- Download the dataset using the `opendatasets` Python library

Here's some sample code for downloading the US Elections Dataset:

```
import opendatasets as od
dataset_url = 'https://www.kaggle.com/tunguz/us-elections-dataset'
od.download('https://www.kaggle.com/tunguz/us-elections-dataset')
```

You can find a list of recommended datasets here: https://jovian.ml/forum/t/recommended-datasets-for-course-project/11711

Step 2: Perform data preparation & cleaning

- Load the dataset into a data frame using Pandas
- Explore the number of rows & columns, ranges of values etc.
- Handle missing, incorrect and invalid data
- Perform any additional steps (parsing dates, creating additional columns, merging multiple dataset etc.)

Step 3: Perform exploratory analysis & visualization

- Compute the mean, sum, range and other interesting statistics for numeric columns
- Explore distributions of numeric columns using histograms etc.
- Explore relationship between columns using scatter plots, bar charts etc.
- Make a note of interesting insights from the exploratory analysis

Step 4: Ask & answer questions about the data

- Ask at least 4 interesting questions about your dataset
- Answer the questions either by computing the results using Numpy/Pandas or by plotting graphs using Matplotlib/Seaborn
- Create new columns, merge multiple dataset and perform grouping/aggregation wherever necessary
- Wherever you're using a library function from Pandas/Numpy/Matplotlib etc. explain briefly what it does

Step 5: Summarize your inferences & write a conclusion

- Write a summary of what you've learned from the analysis
- Include interesting insights and graphs from previous sections
- Share ideas for future work on the same topic using other relevant datasets
- Share links to resources you found useful during your analysis

Step 6: Make a submission & share your work

- Upload your notebook to your Jovian.ml profile using `jovian.commit`.
- **Make a submission here**: https://jovian.ml/learn/data-analysis-with-python-zero-to-pandas/assignment/course-project
- Share your work on the forum: https://jovian.ml/forum/t/course-project-on-exploratory-data-analysis-discuss-and-share-your-work/11684
- Browse through projects shared by other participants and give feedback

(Optional) Step 7: Write a blog post

- A blog post is a great way to present and showcase your work.
- Sign up on Medium.com to write a blog post for your project.
- Copy over the explanations from your Jupyter notebook into your blog post, and embed code cells & outputs
- Check out the Jovian.ml Medium publication for inspiration: https://medium.com/jovianml

Example Projects

Refer to these projects for inspiration:

- Analyzing StackOverflow Developer Survey Results
- Analyzing Covid-19 data using Pandas
- Analyzing your browser history using Pandas & Seaborn by Kartik Godawat
- WhatsApp Chat Data Analysis by Prajwal Prashanth
- Understanding the Gender Divide in Data Science Roles by Aakanksha N S
- 2019 State of Javscript Survey Results
- 2020 Stack Overflow Developer Survey Results

NOTE: Remove this cell containing the instructions before making your submission. You can do using the "Edit > Delete Cells" menu option.

Project Title -Data Analysis on emission of co2 per person in world

explanation - in this course project we will analysed the data which we taken from kaggle . the data tells about the emission of co2 per person in each country by the help of different libraries like pandas ,numpy ,matplotlib,and seaborne

How to run the code

This is an executable *Jupyter notebook* hosted on Jovian.ml, a platform for sharing data science projects. You can run and experiment with the code in a couple of ways: *using free online resources* (recommended) or *on your own computer*.

Option 1: Running using free online resources (1-click, recommended)

The easiest way to start executing this notebook is to click the "Run" button at the top of this page, and select "Run on Binder". This will run the notebook on mybinder.org, a free online service for running Jupyter notebooks. You can also select "Run on Colab" or "Run on Kaggle".

Option 2: Running on your computer locally

1. Install Conda by following these instructions. Add Conda binaries to your system `PATH`, so you can use the `conda` command on your terminal.
2. Create a Conda environment and install the required libraries by running these commands on the terminal:

```
conda create -n zerotopandas -y python=3.8
conda activate zerotopandas
pip install jovian jupyter numpy pandas matplotlib seaborn opendatasets --upgrade
```

3. Press the "Clone" button above to copy the command for downloading the notebook, and run it on the terminal. This will create a new directory and download the notebook. The command will look something like this:

```
jovian clone notebook-owner/notebook-id
```

4. Enter the newly created directory using `cd directory-name` and start the Jupyter notebook.

```
jupyter notebook
```

You can now access Jupyter's web interface by clicking the link that shows up on the terminal or by visiting http://localhost:8888 on your browser. Click on the notebook file (it has a `.ipynb` extension) to open it.

Downloading the Dataset

explanation - we taken data from kaggle in further steps we see how to pull data ,anaylsed it ,and send again

```
!pip install jovian opendatasets --upgrade --quiet
```

Let's begin by downloading the data, and listing the files within the dataset.

how to read a local file(c.s.v) into notebook using pandas

first of all download the c.s.v(or anyother format) file in your local storage any where from various sources(I downloaded from kaggle) then -

open your jupitor notebook click on file then click on open then click on upload section select the file and upload

read the file in notebook using pd.read_csv('name_of_the file')

as i have done

```
# Change this
import pandas as pd
climate_df=pd.read_csv('co2_emissions_tonnes_per_person.csv')
```

```
climate_df
```

	geo	1800	1801	1802	1803	1804	1805	1806	1807	1808	...	2005	2006	2007	2008	
0	Afghanistan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.0529	0.0637	0.0854	0.154	0
1	Albania	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.3800	1.2800	1.3000	1.460	1
2	Algeria	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	3.2200	2.9900	3.1900	3.160	3
3	Andorra	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	7.3000	6.7500	6.5200	6.430	6
4	Angola	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.9800	1.1000	1.2000	1.180	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
187	Venezuela	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	6.1600	6.2200	5.8100	6.360	6
188	Vietnam	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.1600	1.2100	1.2200	1.360	1
189	Yemen	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.9740	1.0100	0.9640	0.999	1
190	Zambia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.1900	0.1850	0.1520	0.166	0
191	Zimbabwe	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.8320	0.7960	0.7420	0.573	0

192 rows × 216 columns

The dataset has been downloaded and extracted.

lets slicing in data using iloc function

```
cl_df=climate_df.iloc[:,201:216]
```

```
cl_df
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2
0	0.0385	0.039	0.0487	0.0518	0.0394	0.0529	0.0637	0.0854	0.154	0.242	0.294	0.412	0.350	0.316	0
1	0.9680	1.030	1.2000	1.3800	1.3400	1.3800	1.2800	1.3000	1.460	1.480	1.560	1.790	1.680	1.730	1
2	2.8200	2.670	2.8100	2.8300	2.7000	3.2200	2.9900	3.1900	3.160	3.420	3.300	3.290	3.460	3.510	3
3	8.0200	7.790	7.5900	7.3200	7.3600	7.3000	6.7500	6.5200	6.430	6.120	6.120	5.870	5.920	5.900	5
4	0.5800	0.573	0.7210	0.4980	0.9960	0.9800	1.1000	1.2000	1.180	1.230	1.240	1.250	1.330	1.250	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
187	6.2200	6.920	7.6100	7.4300	5.7600	6.1600	6.2200	5.8100	6.360	6.290	6.510	6.000	6.650	6.070	6
188	0.6680	0.754	0.8640	0.9520	1.0800	1.1600	1.2100	1.2200	1.360	1.470	1.610	1.700	1.570	1.610	1
189	0.8190	0.881	0.8330	0.8890	0.9430	0.9740	1.0100	0.9640	0.999	1.070	0.993	0.811	0.749	0.997	0
190	0.1730	0.176	0.1780	0.1850	0.1830	0.1900	0.1850	0.1520	0.166	0.186	0.194	0.206	0.249	0.261	0
191	1.1400	1.020	0.9570	0.8430	0.7420	0.8320	0.7960	0.7420	0.573	0.406	0.552	0.665	0.530	0.776	0

192 rows × 15 columns

```
mr_df=climate_df['geo']
```

```
mr_df
```

```
0      Afghanistan
1          Albania
2          Algeria
3          Andorra
4           Angola
          ...
187      Venezuela
188        Vietnam
189          Yemen
190         Zambia
191       Zimbabwe
Name: geo, Length: 192, dtype: object
```

```
result = pd.concat([cl_df, mr_df], axis=1)
```

```
result
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2
0	0.0385	0.039	0.0487	0.0518	0.0394	0.0529	0.0637	0.0854	0.154	0.242	0.294	0.412	0.350	0.316	0
1	0.9680	1.030	1.2000	1.3800	1.3400	1.3800	1.2800	1.3000	1.460	1.480	1.560	1.790	1.680	1.730	1
2	2.8200	2.670	2.8100	2.8300	2.7000	3.2200	2.9900	3.1900	3.160	3.420	3.300	3.290	3.460	3.510	3
3	8.0200	7.790	7.5900	7.3200	7.3600	7.3000	6.7500	6.5200	6.430	6.120	6.120	5.870	5.920	5.900	5
4	0.5800	0.573	0.7210	0.4980	0.9960	0.9800	1.1000	1.2000	1.180	1.230	1.240	1.250	1.330	1.250	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
187	6.2200	6.920	7.6100	7.4300	5.7600	6.1600	6.2200	5.8100	6.360	6.290	6.510	6.000	6.650	6.070	6
188	0.6680	0.754	0.8640	0.9520	1.0800	1.1600	1.2100	1.2200	1.360	1.470	1.610	1.700	1.570	1.610	1
189	0.8190	0.881	0.8330	0.8890	0.9430	0.9740	1.0100	0.9640	0.999	1.070	0.993	0.811	0.749	0.997	0
190	0.1730	0.176	0.1780	0.1850	0.1830	0.1900	0.1850	0.1520	0.166	0.186	0.194	0.206	0.249	0.261	0
191	1.1400	1.020	0.9570	0.8430	0.7420	0.8320	0.7960	0.7420	0.573	0.406	0.552	0.665	0.530	0.776	0

192 rows × 16 columns

we take the data set from 2000 to 2014 which show us the actual figure

```
result.at[75,'geo']
```

```
'India'
```

```
result.at[75,'2014']
```

```
1.73
```

```

```

so result file show the co2 emission per person(in tonnes) in respective countries

Let us save and upload our work to Jovian before continuing.

```
project_name = "zerotopandas-course-project-satwik"
```

```
!pip install jovian --upgrade -q
```

```
import jovian
```

```
jovian.commit(project=project_name)
```

```
[jovian] Updating notebook "kumarsatwik25/zerotopandas-course-project-satwik" on
https://jovian.ai
```

```
[jovian] Committed successfully! https://jovian.ai/kumarsatwik25/zerotopandas-course-project-satwik
```

```
'https://jovian.ai/kumarsatwik25/zerotopandas-course-project-satwik'
```

#Data Preparation and Cleaning

explanation - lets first see the data and prepare it for better understanding

```
result
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2
0	0.0385	0.039	0.0487	0.0518	0.0394	0.0529	0.0637	0.0854	0.154	0.242	0.294	0.412	0.350	0.316	0
1	0.9680	1.030	1.2000	1.3800	1.3400	1.3800	1.2800	1.3000	1.460	1.480	1.560	1.790	1.680	1.730	1
2	2.8200	2.670	2.8100	2.8300	2.7000	3.2200	2.9900	3.1900	3.160	3.420	3.300	3.290	3.460	3.510	3
3	8.0200	7.790	7.5900	7.3200	7.3600	7.3000	6.7500	6.5200	6.430	6.120	6.120	5.870	5.920	5.900	5
4	0.5800	0.573	0.7210	0.4980	0.9960	0.9800	1.1000	1.2000	1.180	1.230	1.240	1.250	1.330	1.250	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
187	6.2200	6.920	7.6100	7.4300	5.7600	6.1600	6.2200	5.8100	6.360	6.290	6.510	6.000	6.650	6.070	6
188	0.6680	0.754	0.8640	0.9520	1.0800	1.1600	1.2100	1.2200	1.360	1.470	1.610	1.700	1.570	1.610	1
189	0.8190	0.881	0.8330	0.8890	0.9430	0.9740	1.0100	0.9640	0.999	1.070	0.993	0.811	0.749	0.997	0
190	0.1730	0.176	0.1780	0.1850	0.1830	0.1900	0.1850	0.1520	0.166	0.186	0.194	0.206	0.249	0.261	0
191	1.1400	1.020	0.9570	0.8430	0.7420	0.8320	0.7960	0.7420	0.573	0.406	0.552	0.665	0.530	0.776	0

192 rows × 16 columns

```
result.shape
```

```
(192, 16)
```

```
result.columns
```

```
Index(['2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008',
       '2009', '2010', '2011', '2012', '2013', '2014', 'geo'],
      dtype='object')
```

```
result
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2
0	0.0385	0.039	0.0487	0.0518	0.0394	0.0529	0.0637	0.0854	0.154	0.242	0.294	0.412	0.350	0.316	0
1	0.9680	1.030	1.2000	1.3800	1.3400	1.3800	1.2800	1.3000	1.460	1.480	1.560	1.790	1.680	1.730	1
2	2.8200	2.670	2.8100	2.8300	2.7000	3.2200	2.9900	3.1900	3.160	3.420	3.300	3.290	3.460	3.510	3
3	8.0200	7.790	7.5900	7.3200	7.3600	7.3000	6.7500	6.5200	6.430	6.120	6.120	5.870	5.920	5.900	5
4	0.5800	0.573	0.7210	0.4980	0.9960	0.9800	1.1000	1.2000	1.180	1.230	1.240	1.250	1.330	1.250	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2
187	6.2200	6.920	7.6100	7.4300	5.7600	6.1600	6.2200	5.8100	6.360	6.290	6.510	6.000	6.650	6.070	6
188	0.6680	0.754	0.8640	0.9520	1.0800	1.1600	1.2100	1.2200	1.360	1.470	1.610	1.700	1.570	1.610	1
189	0.8190	0.881	0.8330	0.8890	0.9430	0.9740	1.0100	0.9640	0.999	1.070	0.993	0.811	0.749	0.997	0
190	0.1730	0.176	0.1780	0.1850	0.1830	0.1900	0.1850	0.1520	0.166	0.186	0.194	0.206	0.249	0.261	0
191	1.1400	1.020	0.9570	0.8430	0.7420	0.8320	0.7960	0.7420	0.573	0.406	0.552	0.665	0.530	0.776	0

192 rows × 16 columns

lets extract some least developing and some developed countries of world

```
x=result.sort_values('2014', ascending=False).head(15)# take the countries which emitte
```

```
x
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	
138	58.60	67.10	63.40	60.30	56.6	58.9	62.8	53.2	46.7	43.5	40.7	41.2	44.6	37.8	45.4	(
174	18.80	20.70	22.40	24.90	25.9	29.4	32.7	34.7	33.7	34.0	36.1	35.2	33.9	34.5	34.2	Trinidac To
88	26.10	27.30	27.20	27.60	28.8	31.4	31.0	30.1	31.2	31.0	29.9	28.5	30.1	27.3	25.2	Ku
12	28.10	20.00	21.40	21.10	21.1	21.6	19.6	25.9	26.7	23.8	23.6	22.4	20.5	23.8	23.4	Ba
181	35.70	30.50	24.20	28.60	27.7	25.4	23.6	22.4	22.8	21.9	19.4	19.1	19.8	19.0	23.3	United Emi
24	14.10	13.30	12.60	13.00	13.9	13.7	13.1	22.5	24.0	20.5	21.1	24.6	24.2	19.2	22.1	E
144	14.30	14.00	14.90	14.50	17.0	16.6	17.6	15.4	16.6	17.6	18.9	17.7	19.4	18.1	19.5	Saudi A
98	18.90	20.00	21.20	22.10	24.9	25.2	24.4	23.2	22.5	20.9	21.6	21.0	20.0	18.5	17.4	Luxemb
183	20.20	19.60	19.60	19.60	19.7	19.6	19.1	19.3	18.5	17.2	17.5	17.0	16.3	16.4	16.5	United S
8	17.30	16.90	17.50	17.10	17.2	17.3	17.8	17.8	18.1	18.2	17.7	17.4	17.0	16.1	15.4	Aus
127	9.65	8.84	10.90	13.60	11.4	11.9	15.3	16.4	15.5	14.3	15.6	16.7	17.1	16.5	15.4	(
30	17.40	17.00	16.60	17.50	17.3	17.3	16.7	16.8	16.8	15.9	15.6	15.6	14.8	14.7	15.1	Ca
55	10.60	11.20	10.80	12.50	12.7	12.4	12.0	13.9	13.1	10.9	13.6	14.0	13.3	15.1	14.8	Es
85	7.84	8.77	8.85	9.34	11.2	11.4	12.3	14.0	14.5	13.2	15.2	15.6	14.4	15.3	14.2	Kazakl
177	8.31	8.34	8.67	9.58	9.9	10.2	10.3	11.5	11.5	10.1	11.3	12.1	12.3	12.4	12.5	Turkmen

```
y_new=x['2014']
```

```
y=result.sort_values('2014', ascending=True).head(15)
# lowest countries with emission
```

```
y
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	
27	0.0424	0.0313	0.0316	0.0232	0.0276	0.0208	0.0244	0.0236	0.0232	0.0225	0.0243	0.0268	0.0303	0

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	
153	0.0533	0.0541	0.0614	0.0604	0.0587	0.0571	0.0554	0.0551	0.0529	0.0510	0.0508	0.0488	0.0477	0
33	0.0211	0.0199	0.0187	0.0408	0.0389	0.0397	0.0391	0.0429	0.0458	0.0427	0.0435	0.0439	0.0482	0
38	0.0173	0.0173	0.0186	0.0192	0.0226	0.0274	0.0285	0.0301	0.0308	0.0280	0.0313	0.0374	0.0348	0
32	0.0713	0.0641	0.0629	0.0589	0.0579	0.0568	0.0593	0.0592	0.0582	0.0575	0.0593	0.0623	0.0653	0
141	0.0658	0.0638	0.0623	0.0600	0.0599	0.0587	0.0574	0.0590	0.0559	0.0577	0.0576	0.0631	0.0683	0
101	0.0764	0.0740	0.0708	0.0743	0.0735	0.0672	0.0680	0.0641	0.0740	0.0643	0.0754	0.0756	0.0681	0
104	0.0749	0.0734	0.0728	0.0703	0.0707	0.0702	0.0712	0.0737	0.0757	0.0552	0.0640	0.0673	0.0621	0
123	0.0614	0.0558	0.0574	0.0600	0.0620	0.0525	0.0490	0.0493	0.0532	0.0612	0.0714	0.0778	0.1050	0
56	0.0534	0.0635	0.0641	0.0688	0.0710	0.0667	0.0698	0.0742	0.0791	0.0776	0.0751	0.0858	0.0926	0
100	0.1190	0.1070	0.0737	0.0985	0.1020	0.0950	0.0891	0.0934	0.0944	0.0861	0.0926	0.1080	0.1210	0
156	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.1230	0
179	0.0595	0.0608	0.0606	0.0600	0.0630	0.0761	0.0859	0.0945	0.1010	0.1030	0.1160	0.1220	0.1120	0
54	0.1790	0.1800	0.1670	0.1940	0.2000	0.1930	0.1380	0.1400	0.0979	0.1190	0.1170	0.1330	0.1450	0
69	0.1180	0.1190	0.1190	0.1470	0.1490	0.1540	0.1530	0.1600	0.1540	0.1550	0.1530	0.1540	0.1540	0

```
x_new=y["2014"]
```

```
x_new
```

```
27     0.0445
153    0.0450
33     0.0538
38     0.0634
32     0.0666
141    0.0740
101    0.0748
104    0.0832
123    0.1110
56     0.1190
100    0.1300
156    0.1300
179    0.1350
54     0.1470
69     0.1570
Name: 2014, dtype: float64
```

Exploratory Analysis and Visualization

explanation -now we have the new_df data which show the co2 emission(per tonnes) by each countries lets analysis this data using beautiful graphs and charts

Let's begin by importing `matplotlib.pyplot` and `seaborn` .

import seaborn as sns import matplotlib import matplotlib.pyplot as plt %matplotlib inline sns.set_style('darkgrid') matplotlib.rcParams['font.size'] = 14 matplotlib.rcParams['figure.figsize'] = (9, 5) matplotlib.rcParams['figure.facecolor'] = '#00000000'

```
dd_ax=x.geo
```

```
dd_ax
```

```
138                   Qatar
174     Trinidad and Tobago
88                   Kuwait
12                  Bahrain
181    United Arab Emirates
24                   Brunei
144            Saudi Arabia
98               Luxembourg
183           United States
8                 Australia
127                    Oman
30                   Canada
55                  Estonia
85               Kazakhstan
177            Turkmenistan
Name: geo, dtype: object
```

explaination - make two parts

```
dg_ax=y.geo
```

```
dg_ax
```

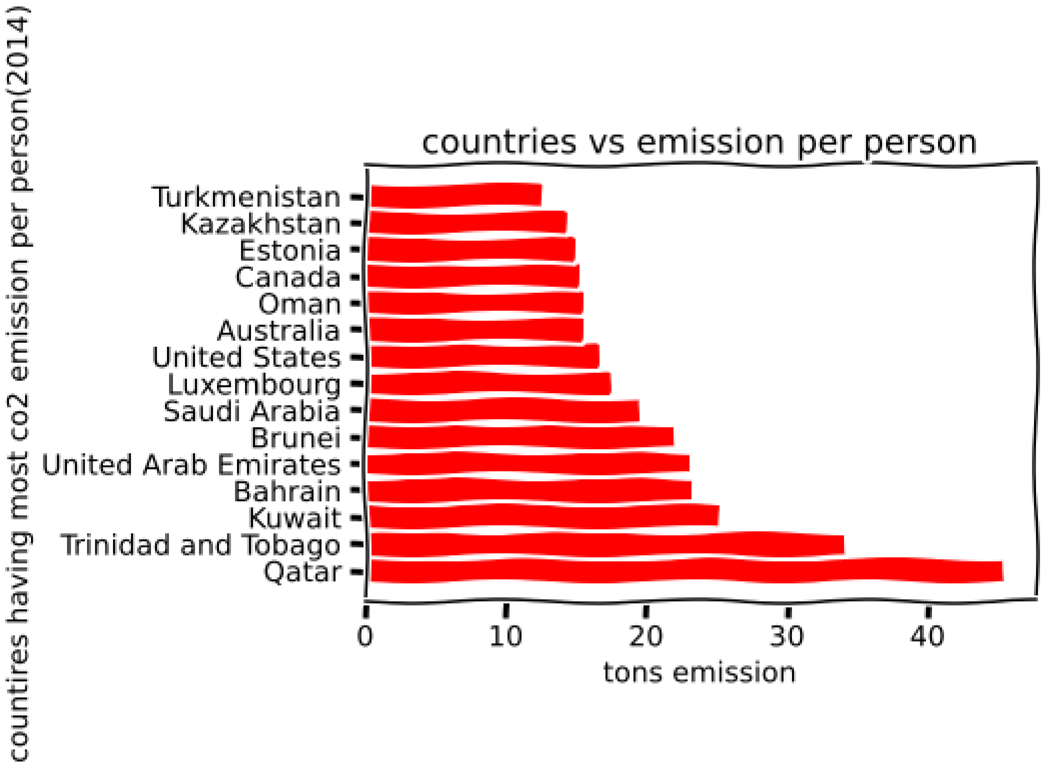
```
27                      Burundi
153                     Somalia
33                         Chad
38               Congo, Dem. Rep.
32     Central African Republic
141                      Rwanda
101                      Malawi
104                        Mali
123                       Niger
56                     Ethiopia
100                  Madagascar
156                 South Sudan
179                      Uganda
54                      Eritrea
69                Guinea-Bissau
Name: geo, dtype: object
```

```
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.barh(dd_ax,y_new,color='r')
plt.xlabel('tons emission')
plt.ylabel('countires having most co2 emission per person(2014)')
plt.title('countries vs emission per person')
plt.xkcd()
# we use plt.xkcd() method for this font(like hand written)
```

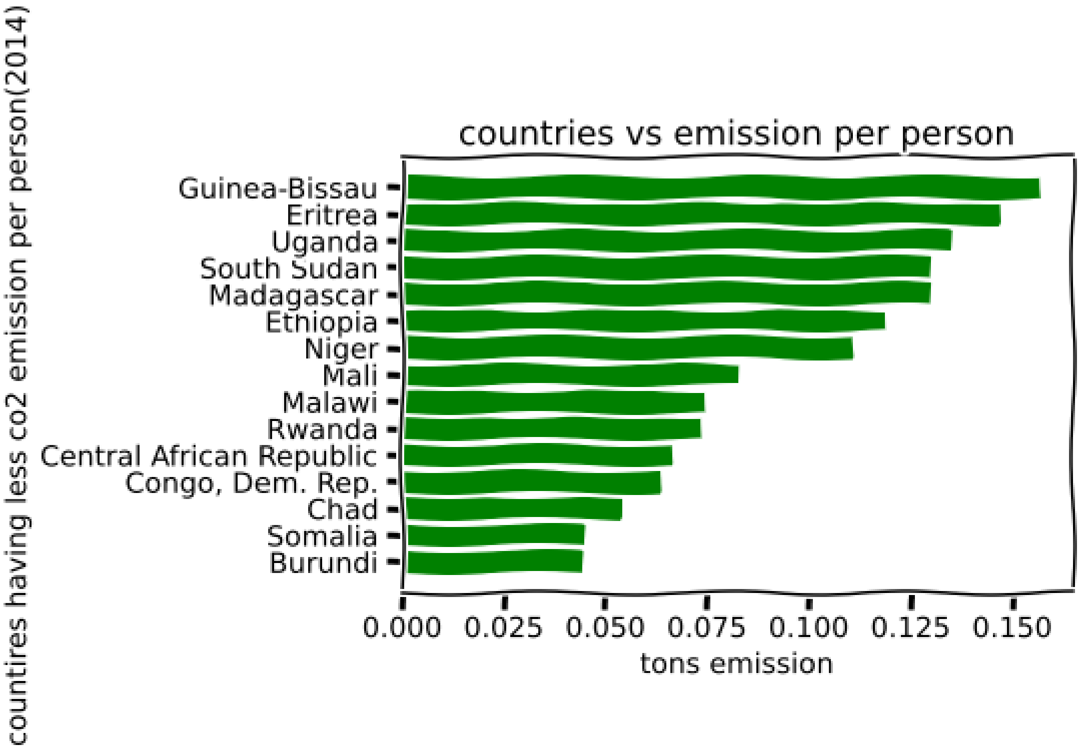
```
<matplotlib.pyplot._xkcd at 0x7f784a1a00d0>
```

```
findfont: Font family ['xkcd', 'xkcd Script', 'Humor Sans', 'Comic Neue', 'Comic Sans
MS'] not found. Falling back to DejaVu Sans.
findfont: Font family ['xkcd', 'xkcd Script', 'Humor Sans', 'Comic Neue', 'Comic Sans
MS'] not found. Falling back to DejaVu Sans.
```



```
plt.barh(dg_ax,x_new,color='g')
plt.xlabel('tons emission')
plt.ylabel('countires having less co2 emission per person(2014)')
plt.title('countries vs emission per person')
plt.xkcd()
```

```
<matplotlib.pyplot._xkcd at 0x7f784a0b8f70>
```



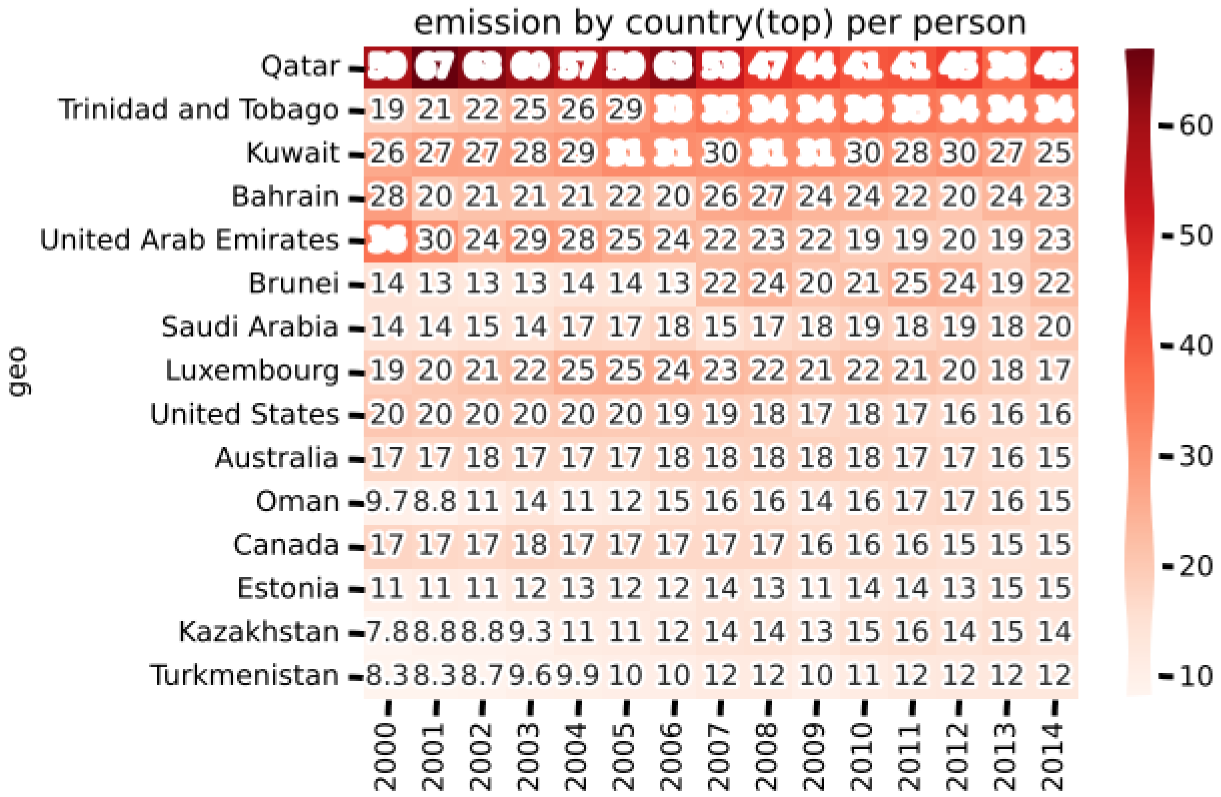
```
xn=x.set_index('geo')
```

```
xn
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014
geo															
Qatar	58.60	67.10	63.40	60.30	56.6	58.9	62.8	53.2	46.7	43.5	40.7	41.2	44.6	37.8	45.4
Trinidad and Tobago	18.80	20.70	22.40	24.90	25.9	29.4	32.7	34.7	33.7	34.0	36.1	35.2	33.9	34.5	34.2
Kuwait	26.10	27.30	27.20	27.60	28.8	31.4	31.0	30.1	31.2	31.0	29.9	28.5	30.1	27.3	25.2
Bahrain	28.10	20.00	21.40	21.10	21.1	21.6	19.6	25.9	26.7	23.8	23.6	22.4	20.5	23.8	23.4
United Arab Emirates	35.70	30.50	24.20	28.60	27.7	25.4	23.6	22.4	22.8	21.9	19.4	19.1	19.8	19.0	23.3
Brunei	14.10	13.30	12.60	13.00	13.9	13.7	13.1	22.5	24.0	20.5	21.1	24.6	24.2	19.2	22.1
Saudi Arabia	14.30	14.00	14.90	14.50	17.0	16.6	17.6	15.4	16.6	17.6	18.9	17.7	19.4	18.1	19.5
Luxembourg	18.90	20.00	21.20	22.10	24.9	25.2	24.4	23.2	22.5	20.9	21.6	21.0	20.0	18.5	17.4
United States	20.20	19.60	19.60	19.60	19.7	19.6	19.1	19.3	18.5	17.2	17.5	17.0	16.3	16.4	16.5
Australia	17.30	16.90	17.50	17.10	17.2	17.3	17.8	17.8	18.1	18.2	17.7	17.4	17.0	16.1	15.4
Oman	9.65	8.84	10.90	13.60	11.4	11.9	15.3	16.4	15.5	14.3	15.6	16.7	17.1	16.5	15.4
Canada	17.40	17.00	16.60	17.50	17.3	17.3	16.7	16.8	16.8	15.9	15.6	15.6	14.8	14.7	15.1
Estonia	10.60	11.20	10.80	12.50	12.7	12.4	12.0	13.9	13.1	10.9	13.6	14.0	13.3	15.1	14.8
Kazakhstan	7.84	8.77	8.85	9.34	11.2	11.4	12.3	14.0	14.5	13.2	15.2	15.6	14.4	15.3	14.2
Turkmenistan	8.31	8.34	8.67	9.58	9.9	10.2	10.3	11.5	11.5	10.1	11.3	12.1	12.3	12.4	12.5

```
plt.figure(figsize=(8,6))
plt.title('emission by country(top) per person')
```

```
sns.heatmap(xn,annot=True,cmap='Reds');
```



```
ny=y.set_index('geo')
```

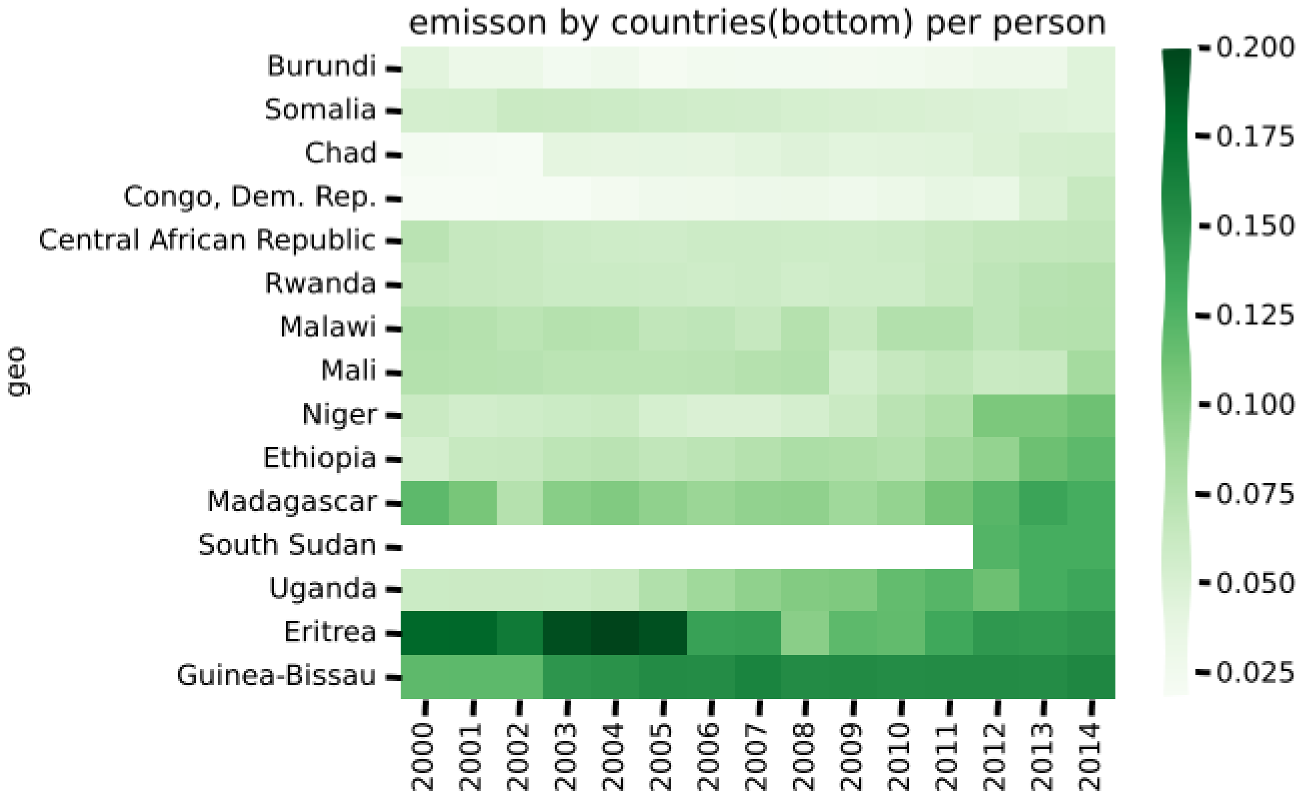
```
ny
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	
geo													
Burundi	0.0424	0.0313	0.0316	0.0232	0.0276	0.0208	0.0244	0.0236	0.0232	0.0225	0.0243	0.0268	0.
Somalia	0.0533	0.0541	0.0614	0.0604	0.0587	0.0571	0.0554	0.0551	0.0529	0.0510	0.0508	0.0488	0.
Chad	0.0211	0.0199	0.0187	0.0408	0.0389	0.0397	0.0391	0.0429	0.0458	0.0427	0.0435	0.0439	0.
Congo, Dem. Rep.	0.0173	0.0173	0.0186	0.0192	0.0226	0.0274	0.0285	0.0301	0.0308	0.0280	0.0313	0.0374	0.
Central African Republic	0.0713	0.0641	0.0629	0.0589	0.0579	0.0568	0.0593	0.0592	0.0582	0.0575	0.0593	0.0623	0.
Rwanda	0.0658	0.0638	0.0623	0.0600	0.0599	0.0587	0.0574	0.0590	0.0559	0.0577	0.0576	0.0631	0.
Malawi	0.0764	0.0740	0.0708	0.0743	0.0735	0.0672	0.0680	0.0641	0.0740	0.0643	0.0754	0.0756	0.
Mali	0.0749	0.0734	0.0728	0.0703	0.0707	0.0702	0.0712	0.0737	0.0757	0.0552	0.0640	0.0673	0.
Niger	0.0614	0.0558	0.0574	0.0600	0.0620	0.0525	0.0490	0.0493	0.0532	0.0612	0.0714	0.0778	0.
Ethiopia	0.0534	0.0635	0.0641	0.0688	0.0710	0.0667	0.0698	0.0742	0.0791	0.0776	0.0751	0.0858	0.
Madagascar	0.1190	0.1070	0.0737	0.0985	0.1020	0.0950	0.0891	0.0934	0.0944	0.0861	0.0926	0.1080	0.
South Sudan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.
Uganda	0.0595	0.0608	0.0606	0.0600	0.0630	0.0761	0.0859	0.0945	0.1010	0.1030	0.1160	0.1220	0.
Eritrea	0.1790	0.1800	0.1670	0.1940	0.2000	0.1930	0.1380	0.1400	0.0979	0.1190	0.1170	0.1330	0.

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	
geo													
Guinea-Bissau	0.1180	0.1190	0.1190	0.1470	0.1490	0.1540	0.1530	0.1600	0.1540	0.1550	0.1530	0.1540	0.

heatmap - heat map show the best explanation about countries

```
plt.figure(figsize=(8,6))
plt.title("emisson by countries(bottom) per person")
sns.heatmap(ny, cmap='Greens');
```



lets extract the information about india which show the development of person in india because emission per person is good indicator for measure development of person reside inside the country

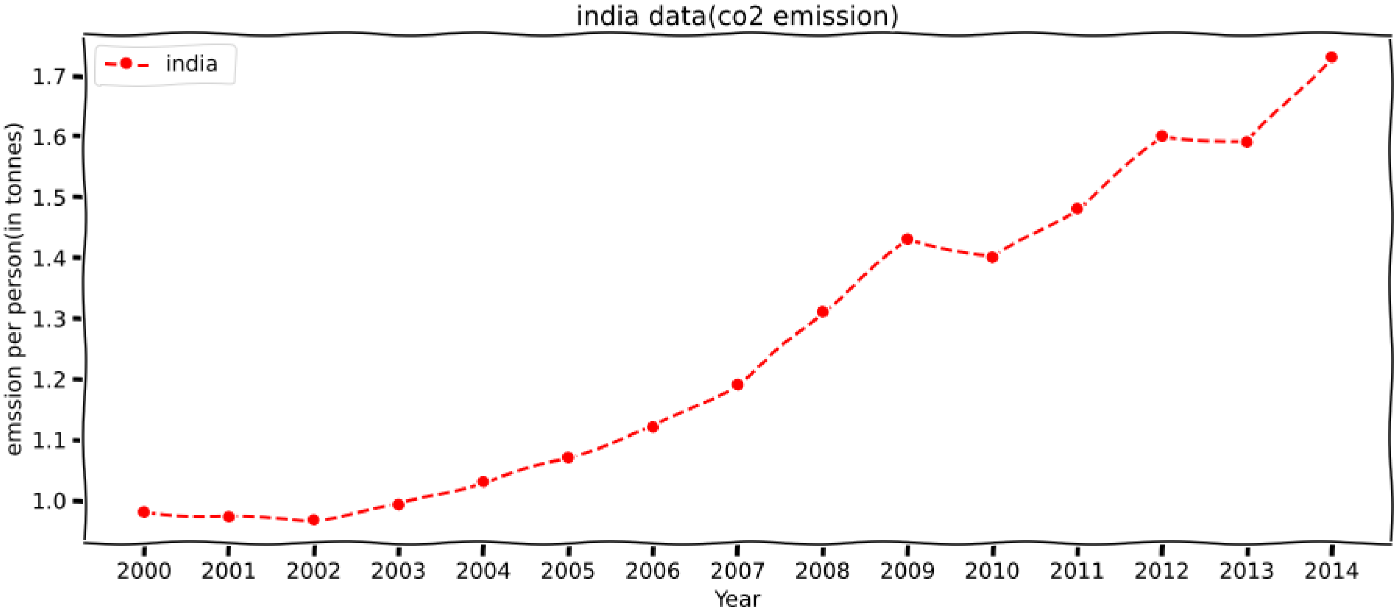
explanation -plot the graph set of india then after that we compare to india with u.s.a

```
new_r=result.set_index('geo')
```

```
plt.figure(figsize=(15,6))
plt.plot(new_r.loc['India'],'o--r')
plt.xlabel('Year')
plt.ylabel('emssion per person(in tonnes)')

plt.title("india data(co2 emission)")

plt.legend(['india ']);
```

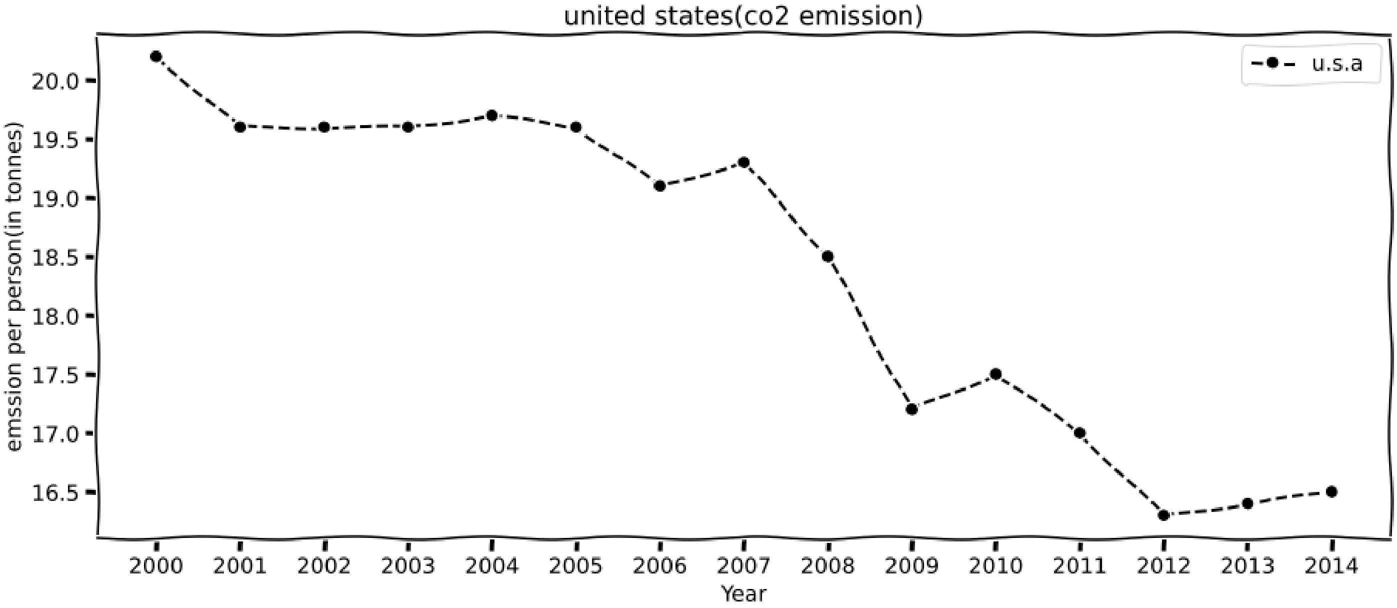


lets take u.s.a data and compare with india data

```
plt.figure(figsize=(15,6))
plt.plot(new_r.loc['United States'],'o--k')
plt.xlabel('Year')
plt.ylabel('emssion per person(in tonnes)')

plt.title("united states(co2 emission)")

plt.legend(['u.s.a ']);
```

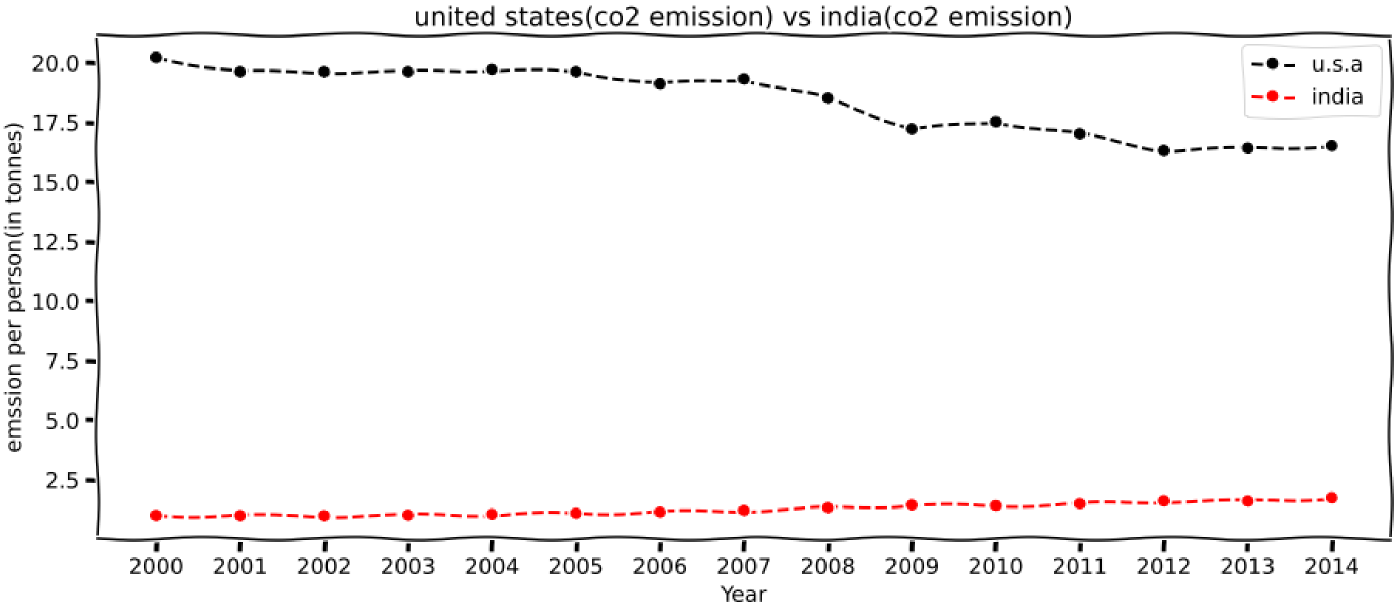


in above graphs we conclude that the emission of india was increase from 2000 to 2014 on the other hand the per person emission is decrease from 2000 to 2014

```
plt.figure(figsize=(15,6))
plt.plot(new_r.loc['United States'],'o--k')
plt.plot(new_r.loc['India'],'o--r')
plt.xlabel('Year')
plt.ylabel('emssion per person(in tonnes)')

plt.title("united states(co2 emission) vs india(co2 emission)")

plt.legend(['u.s.a ','india']);
```



explanation - so above graph show there is huge gap between india and u.s.a in terms of co2 emission per person

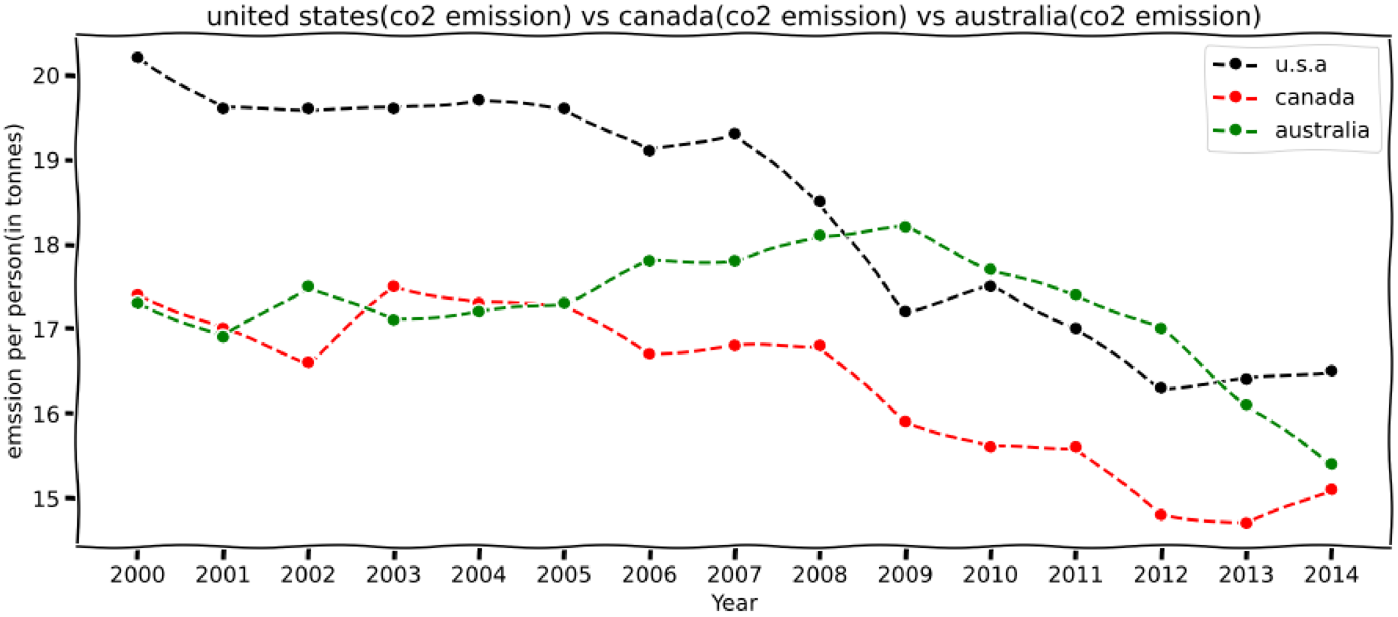
let;s compare some developed country as well

```
plt.figure(figsize=(15,6))
plt.plot(new_r.loc['United States'],'o--k')
plt.plot(new_r.loc['Canada'],'o--r')
plt.plot(new_r.loc['Australia'],'o--g')

plt.xlabel('Year')
plt.ylabel('emssion per person(in tonnes)')

plt.title("united states(co2 emission) vs canada(co2 emission) vs australia(co2 emissic

plt.legend(['u.s.a ','canada','australia']);
```



explanation-if we see the above graph we analysed that u.s.a would do great work for controlling the emission

Let us save and upload our work to Jovian before continuing

```python
import jovian
```

```python
jovian.commit()
```

```
[jovian] Updating notebook "kumarsatwik25/zerotopandas-course-project-satwik" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/kumarsatwik25/zerotopandas-course-project-satwik
```

```
'https://jovian.ai/kumarsatwik25/zerotopandas-course-project-satwik'
```

Asking and Answering Questions

now we put some questions and with the help of graphs and data try to answer those questions

Q1: - which country has most emission in recent years

```python
# which country has most emission
result.sort_values('2014', ascending=False).head(5)
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	geo
138	58.6	67.1	63.4	60.3	56.6	58.9	62.8	53.2	46.7	43.5	40.7	41.2	44.6	37.8	45.4	Qatar
174	18.8	20.7	22.4	24.9	25.9	29.4	32.7	34.7	33.7	34.0	36.1	35.2	33.9	34.5	34.2	Trinidad and Tobago
88	26.1	27.3	27.2	27.6	28.8	31.4	31.0	30.1	31.2	31.0	29.9	28.5	30.1	27.3	25.2	Kuwait
12	28.1	20.0	21.4	21.1	21.1	21.6	19.6	25.9	26.7	23.8	23.6	22.4	20.5	23.8	23.4	Bahrain

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	geo
181	35.7	30.5	24.2	28.6	27.7	25.4	23.6	22.4	22.8	21.9	19.4	19.1	19.8	19.0	23.3	United Arab Emirates

for more clarification see the graphs above

Q2: - which countries has lowest emission per person in recent years

```
# which countries has lowest emission per person in recent years
result.sort_values('2014', ascending=True).head(5)
```

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	
27	0.0424	0.0313	0.0316	0.0232	0.0276	0.0208	0.0244	0.0236	0.0232	0.0225	0.0243	0.0268	0.0303	0
153	0.0533	0.0541	0.0614	0.0604	0.0587	0.0571	0.0554	0.0551	0.0529	0.0510	0.0508	0.0488	0.0477	0
33	0.0211	0.0199	0.0187	0.0408	0.0389	0.0397	0.0391	0.0429	0.0458	0.0427	0.0435	0.0439	0.0482	0
38	0.0173	0.0173	0.0186	0.0192	0.0226	0.0274	0.0285	0.0301	0.0308	0.0280	0.0313	0.0374	0.0348	0
32	0.0713	0.0641	0.0629	0.0589	0.0579	0.0568	0.0593	0.0592	0.0582	0.0575	0.0593	0.0623	0.0653	0

for more clarification see the graphs and bar above

Q3: - total countries in world till 2014

```
result.describe()
```

	2000	2001	2002	2003	2004	2005	2006	2007	
count	189.000000	189.000000	190.000000	190.000000	190.000000	190.000000	190.000000	191.000000	191.
mean	4.442050	4.507479	4.449909	4.570025	4.609213	4.632572	4.696715	4.700221	4.
std	6.704058	6.896657	6.642406	6.677873	6.586254	6.738075	6.943929	6.706373	6.
min	0.017300	0.017300	0.018600	0.019200	0.022600	0.020800	0.024400	0.023600	0.
25%	0.497000	0.529000	0.547000	0.512500	0.552750	0.630250	0.611500	0.595500	0.
50%	2.020000	2.130000	2.030000	2.095000	2.140000	2.140000	2.210000	2.230000	2.
75%	6.080000	6.300000	6.430000	6.407500	6.487500	6.545000	6.425000	6.435000	6.
max	58.600000	67.100000	63.400000	60.300000	56.600000	58.900000	62.800000	53.200000	46.

```
total_countries_till_2014=192
```

```
print('total countries in 2014 were ',total_countries_till_2014)
```

```
total countries in 2014 were  192
```

Q4: - how many countries were added between 2000 to 2014

```
result.count()
```

```
2000    189
2001    189
2002    190
2003    190
2004    190
2005    190
2006    190
2007    191
2008    191
2009    191
2010    191
2011    191
2012    192
2013    192
2014    192
geo     192
dtype: int64
```

countries in 2000 were 189 in 2014 total countries were 192 so

```
countries_increse=192-189
```

```
print("total countries increase ",countries_increse)
```

```
total countries increase  3
```

go and check out which countries were added after 2000 and before 2014 on google

Q5: - what is columns in data sets and india position

```
result.columns
```

```
Index(['2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008',
       '2009', '2010', '2011', '2012', '2013', '2014', 'geo'],
      dtype='object')
```

```
new_r.loc['India']
```

```
2000    0.980
2001    0.972
2002    0.967
2003    0.992
2004    1.030
2005    1.070
2006    1.120
2007    1.190
2008    1.310
```

```
2009    1.430
2010    1.400
2011    1.480
2012    1.600
2013    1.590
2014    1.730
Name: India, dtype: float64
```

Let us save and upload our work to Jovian before continuing.

```python
import jovian
```

```python
jovian.commit()
```

Inferences and Conclusion

1.the higest emmision of co2 per person is quatar due to more oil wells and resources 2.the lowest emmision of co2 per person is burundi because it is least developing country 3.there is huge gap between the emmision of u.s.a and india 4.the indian emmision were incresed in recent years 5.the u.s.a emmision were reduced in recent years 6.total countries till 2014 were 192

```python
import jovian
```

```python
jovian.commit()
```

```
[jovian] Updating notebook "kumarsatwik25/zerotopandas-course-project-satwik" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/kumarsatwik25/zerotopandas-course-project-satwik
```

```
'https://jovian.ai/kumarsatwik25/zerotopandas-course-project-satwik'
```

References and Future Work

refrences - for matplot and seaborn see-matplotlib corey(you tube channel) for pandas i only see jovian videos and some other sites like w3 school

> Submission Instructions (delete this cell)
>
> - Upload your notebook to your Jovian.ml profile using `jovian.commit`.
> - **Make a submission here:** https://jovian.ml/learn/data-analysis-with-python-zero-to-pandas/assignment/course-project
> - Share your work on the forum: https://jovian.ml/forum/t/course-project-on-exploratory-data-analysis-discuss-and-share-your-work/11684
> - Share your work on social media (Twitter, LinkedIn, Telegram etc.) and tag @JovianML

> (Optional) Write a blog post
>
> - A blog post is a great way to present and showcase your work.
> - Sign up on Medium.com to write a blog post for your project.
> - Copy over the explanations from your Jupyter notebook into your blog post, and embed code cells & outputs
> - Check out the Jovian.ml Medium publication for inspiration: https://medium.com/jovianml

```
import jovian
```

```
jovian.commit()
```