Exploratory Data Analysis Basics

Exploratory Data Analysis (EDA) is a crucial task for researchers and data scientists to gain a deeper understanding of their data. By visually and statistically examining data, EDA helps uncover patterns, identify outliers, detect data quality issues, and reveal hidden relationships between variables. It aids in formulating hypotheses, selecting appropriate modeling techniques, and making informed decisions during subsequent analysis.

In the following paragraphs I break down some important steps and techniques used in EDA including: data cleaning and manipulation, visualization, identifying correlation, and using the split-apply-combine strategy. I chose to demonstrate some of these skills using global C02 emissions data from the World Bank WDI data set (available here: https://datatopics.worldbank.org/world-development-indicators). A more complete explanation of each technique along with code and visualizations can be found in this projects GitHub repository.

Data Cleaning and Manipulation

When I encounter a new data set the first thing I do is get the data into a Pandas dataframe. I'm most comfortable cleaning and manipulating data using Python Pandas library because that is what I have the most experience using. I like to look at the shape of the data first to see how many rows and columns are in the dataframe. I also usually print the column names. I usually print the first few rows or a sample of the data just to see what it looks like too. Here I would be looking to see what data types were included (string, ints, floats) and if different data types were mixed into the same columns. I might also be getting a sense for the number and location of missing values.

The next step I typically perform is to use the Pandas describe method. Here I want to see if the column names and data types make sense. I also want to look at the distribution of data in each column. Are there lots of NaN values or outliers? This might not make sense in all cases. The WDI data set has indicator and year as columns so many indicator types are included in each data column. The Pandas melt and pivot functions can be used to rearrange rows and columns.

It's important to keep in mind that cleaning and manipulating data can be a form of analysis. Decisions about filtering or dropping values can have a big impact on results. For example, if I filter out countries with low CO2 emissions and focus on countries with highest CO2 emissions I might not identify some important correlation between a variable and CO2 emissions that is only apparent when looking at countries with low CO2 emissions. Filtering data or deleting NaN values can also inadvertently cause selection bias. You might accidentally only be looking at a particular subset of the population that has different characteristics than the population as a whole.

The Importance of Visualization

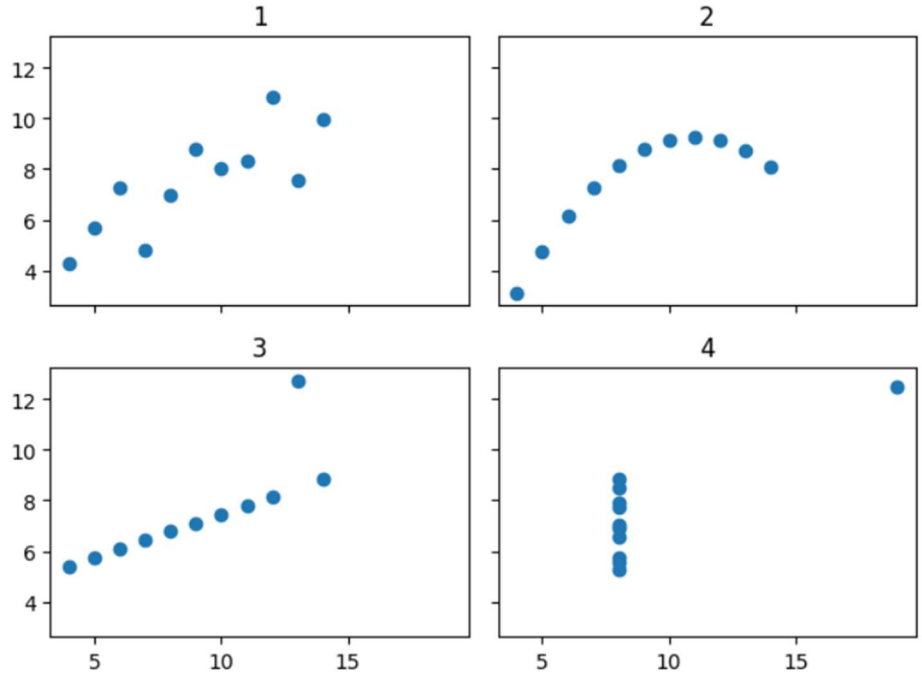
Visualizing data is important in data exploration because it can reveal relationships that descriptive statistics do not. A classic example of this is Anscombe's quartet. Anscombe's was an English statistician that created four data sets that have nearly identical descriptive statistics but appear very different when graphed. Here is the raw data.

	x1	y1	x2	y2	x3	y3	x4	y4
count	11.000	11.000	11.000	11.000	11.000	11.00	11.000	11.000
mean	9.000	7.501	9.000	7.501	9.000	7.50	9.000	7.501
std	3.317	2.032	3.317	2.032	3.317	2.03	3.317	2.031
min	4.000	4.260	4.000	3.100	4.000	5.39	8.000	5.250
25%	6.500	6.315	6.500	6.695	6.500	6.25	8.000	6.170
50%	9.000	7.580	9.000	8.140	9.000	7.11	8.000	7.040
75%	11.500	8.570	11.500	8.950	11.500	7.98	8.000	8.190
max	14.000	10.840	14.000	9.260	14.000	12.74	19.000	12.500

The table below depicts the mean and standard deviation of each raw data column.

	x1	y1	x2	y2	x3	y3	x4	y4
mean	9.000	7.501	9.000	7.501	9.000	7.50	9.000	7.501
std	3.317	2.032	3.317	2.032	3.317	2.03	3.317	2.031

As you can see the mean and standard deviation of each x column is nearly identical. The same is true for each y value column. If we just looked at descriptive statistics, we might conclude that these four sets of x and y values are very similar. Now let's use a scatter plot to visualize the data.



As you can see each data set has a very different distribution that is difficult to notice without visualizing the data.

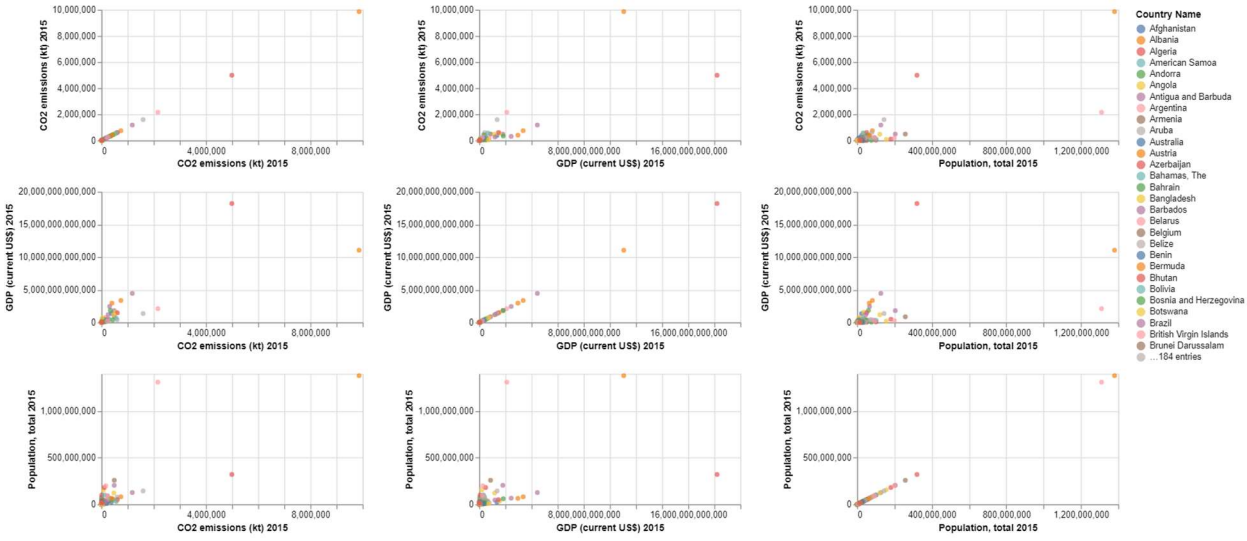
Correlation

What is correlation? If someone told me that two variables were correlated, I would expect that a change in one of the variables would lead to a predictable (positive or negative) change in the other. You could also say that if two variables are correlated then they are statistically related. One way to describe the relatedness of two variables is by calculating the correlation coefficient. The Pearson correlation coefficient is the covariance of two variables divided by the product of their standard deviations. The result is always between -1 and 1. Values closer to 1 or -1 indicate a stronger relationship, more variance in x can be explained by y, and values closer to zero indicate a weaker relationship between variables.

I used the WDI dataset to explore measures of CO2 emissions and other variables that I thought might have some positive or negative relationship with CO2 emissions. I used Pandas .corr method to calculate the Pearson correlation coefficient between pairs of variables to evaluate relatedness of the variables. Here is an example of the output.

	CO2 emissions (kt) 2015
CO2 emissions (kt) 2015	1.000000
GDP (current US$) 2015	0.830580
Population, total 2015	0.807873
Energy use (kg of oil equivalent per capita) 2015	0.129602
Alternative and nuclear energy (% of total energy use) 2015	-0.120053
Fossil fuel energy consumption (% of total) 2015	0.228820

Another way to quickly see if there are any correlated variables in your data is to use a scatter plot matrix which is also known as a splom. Here is an example of a splom using a few variables from the World Bank WDI data.



Split-Apply-Combine

The split-apply-combine technique is useful for many different data manipulation and analysis tasks. Hadley Wickham first described the software pattern that he called “the split-apply-combine strategy” in a 2011 paper published in the Journal of Statistical Software. Wickham described the split-apply-combine technique as breaking up a big problem into manageable pieces, operating on each piece independently, and then putting all the pieces back together. Wickham used R programming language in the paper but could have easily substituted Python. In Pandas the split-apply-combine technique is implemented using the Group by method. Calling df.groupby(df[‘column’]) creates a groupby object. No splitting happens until some aggregation function is called on the groupby object. By default groupby sorts the keys and drops NaN values. Here is an example Pandas groupby to group by indicator then get mean value and a count of values for selected years.

```
new_df= wdi_df[['Indicator Name','2017', '2018', '2019']].groupby('Indicator Name').agg(['mean','count'])
new_df.head(5)
```

	2017		2018		2019	
	mean	count	mean	count	mean	count
Indicator Name						
ARI treatment (% of children under 5 taken to a health provider)	61.158333	12	66.627273	22	66.650000	4
Access to clean fuels and technologies for cooking (% of population)	66.673387	186	67.056183	186	67.432258	186
Access to clean fuels and technologies for cooking, rural (% of rural population)	58.276882	186	58.693548	186	59.115860	186
Access to clean fuels and technologies for cooking, urban (% of urban population)	75.413172	186	75.673118	186	75.918280	186
Access to electricity (% of population)	85.183127	212	85.856512	211	86.361743	212

Here is another example where I am using a specific indicator value to rank each country over selected years.

```
indi = 'CO2 emissions (kt)'
new_df= wdi_df[['Country Name','Indicator Name', '2004', '2016', '2017', '2018','2019']].groupby(
    ['Indicator Name']).get_group(indi).set_index('Country Name').rank(ascending=False, axis=0, numeric_only=True)
print(new_df.sort_values('2019').head())
#new_df
```

```
                  2004  2016  2017  2018  2019
Country Name
China              2.0   1.0   1.0   1.0   1.0
United States      1.0   2.0   2.0   2.0   2.0
India              5.0   3.0   3.0   3.0   3.0
Russian Federation 3.0   4.0   4.0   4.0   4.0
Japan              4.0   5.0   5.0   5.0   5.0
```