Welcome to AVELA's Research Study!

The student's goal is to complete as many tasks as you can. You can use your AI model and the slides provided for support.

Introduction: Data Activism for Climate Change

In this activity, we will analyze recent carbon footprint trends to learn why changes in our climate happen. We will be able determine the impact our carbon footprint has on our environment to help us better understand climate change!

4a. Understanding the Data Before manipulating the data, it's important to understand what it represents and how it applies to the situation or topic being explored.

Why are the numbers decimals? Aren't there more CO2 emissions?

The numbers look "small" because these values actually use really big measuring units (like millions) that would be hard to read and work with in our data frame.

For Example:

The value 0.041 has the unit *million metric tons*, which is the equivalent to filling *470 billion* balloons with CO2!

NOTE: You won't need to understand this conversion to complete the activity; it's enough to understand that larger values correlate to more CO2 emissions *by the millions.*

NaN?

Not a Number (NaN) values are used in this table to represent missing or incomplete data.

NOTE: Negative values in this table pertain to columns we WON'T be analyzing.

Section 2: Data Cleaning

To make the Data Frame easier to understand, we'll remove irrelevant columns and rename the long column names to something shorter.

Task: Create a new DataFrame called `2023_global_emissions` using the `df_CO2_stats` dataframe and clean it.

ISO Code Country Year Total CO2 Emisssions CO2 Emissions per Person CO2 from Oil CO2 from Coal CO2 from Natural Gas

@title ↓ Run this cell to filter the year and only see values from the year 2023.

renaming the columns for the DataFrame to make it easier for indexing the columns

df_2023_emissions = df_CO2_stats[df_CO2_stats["Year"] == 2023]The selected code filters the df_CO2_stats DataFrame to include only the rows where the 'Year' column is equal to 2023.

@title ↓ Run this cell to filter the year and only see values from the year 2023.: This is a title for the cell that can be toggled.

renaming the columns for the DataFrame to make it easier for indexing the columns: This is a comment explaining the purpose of the code, although the renaming of columns happens in the previous cell.

df_2023_emissions = df_CO2_stats[df_CO2_stats["Year"] == 2023]:

This line creates a new DataFrame called df_2023_emissions that contains only the rows from df_CO2_stats where the value in the 'Year' column is 2023.

This code is useful for isolating data from a specific year for further analysis.

df_2023_emissions

The selected code simply displays the contents of the df_2023_emissions DataFrame that was created in the previous cell. This allows you to view the data after it has been filtered to include only the year 2023.

@title ↓ Run this cell to filter out any NaN (Not a Number) values. df_2023_only = df_CO2_stats[df_CO2_stats["Year"] == 2023].dropna() df_2023_only

The selected code filters the df_CO2_stats DataFrame to include only the rows from the year 2023 and also removes any rows that contain missing values (NaN).

@title ↓ Run this cell to filter out any NaN (Not a Number) values.: This is a title for the cell that can be toggled.

df_2023_only = df_CO2_stats[df_CO2_stats["Year"] == 2023].dropna(): This line does two things:

df_CO2_stats[df_CO2_stats["Year"] == 2023]: This part filters the DataFrame to keep only the rows where the 'Year' column is 2023.

.dropna(): This part removes any rows that still have missing values (NaN) after the year filtering.

df_2023_only: This line displays the resulting df_2023_only DataFrame. This code is useful for creating a clean dataset for the year 2023, free from missing data.

Section 4: Creating Visualizations

In this section, you will create a bar graph, pie chart, and heatmap using the data frame we previously created.

NOTE: Go through these at your own pace. It's okay if you don't finish all 3!*

Creating a Bar Graph

CONTEXT:

You will create a bar graph that shows CO_2 emissions from oil for the first 7 countries in the dataset.

This activity uses the `df_2023_emissions` DataFrame. You'll focus on the column `CO2 from Oil`, and apply a different color to each bar to help visually compare how much each country contributed through oil-related emissions. This chart helps us explore the role of oil in global CO_2 emissions and how it varies by country.

df_few = df_2023_only.iloc[0:7, 3]

colors = ['green', 'blue', 'pink', 'orange', 'purple', 'yellow', 'teal']

ax = df_few.plot.bar(x="Country", y="Total CO_2 Emissions", color=colors)

ax.set_xlabel("Country") ax.set_ylabel("CO_2 Emissions") ax.set_title("CO_2 Emissions for the First 7 Countries")

ax.figure.show()

The selected code generates a bar plot showing the total CO2 emissions for the first 7 countries in the df_2023_only DataFrame.

Here's a breakdown:

df_few = df_2023_only.iloc[0:7, 3]: This line creates a new DataFrame df_few by selecting the first 7 rows (0:7) and the column at index 3 (3) from the df_2023_only DataFrame. Based on the previous code, column index 3 corresponds to 'Total CO2 Emissions'.

colors = ['green', 'blue', 'pink', 'orange', 'purple', 'yellow', 'teal']: This line defines a list of colors to be used for the bars in the plot. ax = df_few.plot.bar(x="Country", y="Total CO_2 Emissions", color=colors): This line generates the bar plot: df_few.plot.bar(): Calls the bar plotting function on the df_few DataFrame.

x="Country": Attempts to set the x-axis labels to the 'Country' column. However, since the DataFrame df_few only contains the 'Total CO2 Emissions' column (index 3), this will likely cause an issue or the x-axis will use the DataFrame index. y="Total CO_2 Emissions": Sets the y-axis to the 'Total CO_2 Emissions' column. color=colors: Assigns the specified colors to the bars.

ax.set_xlabel("Country"): Sets the label for the x-axis to "Country".

ax.set_ylabel("CO_2 Emissions"): Sets the label for the y-axis to "CO_2 Emissions".

ax.set_title("CO_2 Emissions for the First 7 Countries"): Sets the title of the bar plot. ax.figure.show(): Displays the generated plot. Based on the output, it seems the x-axis is using the DataFrame index instead of the country names. This is because the df_few DataFrame was created by selecting only the 'Total CO2 Emissions' column.

To fix this, you would need to include the 'Country' column when creating df_few.

Creating a Pie Chart

CONTEXT:

You will create a pie chart that represents 5 different countries and their CO2 emissions from oil in the year 2023.

This will be based on data from our new df_5_countries data frame, and you will visualize this column:

CO2 from Oil

NOTE: Make sure to copy+paste your code to the next cell as you progress through this section. Each cell builds upon eachother!

Task: Create a pie chart using the `df_5_countries` DataFrame to show CO2 emission from oil

Exercise 4) Create a pie chart for the df_5_countries DataFrame showing CO2 emissions from oil

oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil")

oil_emissions_pie

The selected code generates a basic pie chart using the df_5_countries DataFrame to visualize the distribution of "CO2 from Oil" emissions among the five selected countries.

Here's a breakdown:

Exercise 4) Create a pie chart for the df_5_countries DataFrame showing CO2 emissions from oil ''': This is a multi-line comment explaining the purpose of the code cell.

oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil"): This line creates the pie chart. df_5_countries.plot.pie():

This calls the pie plotting function on the df_5_countries DataFrame.

y = "CO2 from Oil": This specifies that the values from the "CO2 from Oil" column should be used to determine the size of each slice in the pie chart. oil_emissions_pie: This line displays the generated pie chart.

This code provides a visual representation of how the total CO2 emissions from oil are divided among the five countries in the df_5_countries DataFrame.

Task Add the percentage of CO2 emissions from oil for each country to your pie chart

Exercise 5 Create a pie chart for the df_5_countries DataFrame showing CO2 emissions from oil

Use the autopct parameter to display the percentage each slice represents

oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil", autopct ='%1.0f%%')

oil_emissions_pie

The selected line # visualize the pie chart is a comment. Comments in Python start with a # and are ignored by the interpreter. They are used to add explanations or notes to the code, making it easier for humans to understand. In this case, the comment indicates that the line below it is intended to display the pie chart that was just created.

The line oil_emissions_pie then displays the pie chart object that was assigned to the variable oil_emissions_pie in the previous line of code within that cell.

Task: Remove the legend from your pie chart.

Exercise 6) Create a pie chart for the df_5_countries DataFrame showing CO2 emissions from oil Use the autopct parameter to display the percentage each slice represents Remove the legend from the chart

oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil", autopct ='%1.0f%%')

oil_emissions_pie.get_legend().remove()

oil_emissions_pie

The selected code creates a pie chart showing CO2 emissions from oil for the five selected countries, displays the percentage each slice represents, and removes the legend.

Here's a breakdown:

Create a pie chart of CO2 emissions from oil using df_5_countries: This is a comment explaining the purpose of the code.

autopct='%1.0f%%' shows percentage labels without decimals: This comment explains the autopct parameter.

oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil", autopct ='%1.0f%%'):

This line creates the pie chart. df_5_countries.plot.pie():

Calls the pie plotting function on the df_5_countries DataFrame.

y = "CO2 from Oil": Specifies that the "CO2 from Oil" column provides the values for the pie slices. autopct ='%1.0f%%':

This parameter formats the percentage labels on each slice. %1.0f means a floating-point number with zero decimal places, and %% adds a literal percentage sign.

visualize the pie chart: This is a comment. oil_emissions_pie.get_legend().remove():

This line removes the legend from the pie chart. get_legend() retrieves the legend object, and .remove() removes it from the plot.

oil_emissions_pie: This line displays the generated pie chart without the legend.

This code is useful for creating a clean and informative pie chart that focuses on the data distribution without the distraction of a legend.

Task: Make finishing touches to your pie chart

Exercise 7) Create a pie chart for the df_5_countries DataFrame for the total for your chosen

Exercise 7) Create a pie chart for the df_5_countries DataFrame for the total for your chosen type of CO2 emission using the autopct parameter to show the percentages of each sector of the chart, removing the legend, and adding a title that describes the type of CO2 emission you chose, ending with "Comparison from 5 Different Countries in 2023"

use the df_wash DataFrame that we created in the first code cell

oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil", autopct = '%1.0f%%', labels = df_5_countries.loc[:, "Country"], ylabel='', title = "CO2 Emissions Comparison from 5 Different Countries in 2023")

make these question marks

remove the legend oil_emissions_pie.get_legend().remove()

visualize the pie chart oil_emissions_pie

The selected code creates a pie chart with several enhancements for better readability.

Here's a breakdown:

The multiline comment at the beginning explains the goal: to create a pie chart of CO2 emissions, show percentages, remove the legend, and add a specific title.

oil_emissions_pie = df_5_countries.plot.pie(...): This line generates the pie chart using the df_5_countries DataFrame.

y = "CO2 from Oil": Specifies that the pie chart should represent the values from the "CO2 from Oil" column.

autopct = '%1.0f%%': Formats the percentage labels on each slice to display as whole numbers followed by a percentage sign.

labels = df_5_countries.loc[:, "Country"]: Uses the "Country" column from the df_5_countries DataFrame as labels for each slice of the pie chart.

ylabel='': Removes the default y-axis label. title = "CO2 Emissions Comparison from 5 Different Countries in 2023": Sets the title of the pie chart.

use the df_wash DataFrame that we created in the first code cell: This is a comment that seems to refer to a different DataFrame name than what is used in the code (df_5_countries).

make these question marks: This is a comment that doesn't correspond to any executable code.

remove the legend: This is a comment explaining the next line. oil_emissions_pie.get_legend().remove(): This line removes the legend from the pie chart object.

visualize the pie chart: This is a comment explaining the next line.

oil_emissions_pie: This line displays the final pie chart with all the specified customizations.

This code effectively creates a clear and informative pie chart visualizing the distribution of oil-related CO2 emissions among the five selected countries.

Creating a Heatmap

CONTEXT:

In this activity, you'll create a heatmap on a world map that shows how much total CO_2 each country emitted in the year 2023.

You'll use the column `Total CO2 Emissions` to color each country based on how much it contributed to global emissions.

The choropleth function is called on the Plotly Express library The first input parameter into the choropleth function is 'df_2023_only' Set locations = "ISO Code", locationmode = "ISO-3", color = "Total CO2 Emissions" The color_continuous_scale can also = "Viridis"

fig2 = px.choropleth(df_2023_only, locations="ISO Code", locationmode="ISO-3", color="Total CO2 Emissions", color_continuous_scale="Turbo", title="Total CO2 Emissions by Country (2023)")

Center the title and show the map fig2.update_layout(title_x=0.5) fig2.show()

The selected code generates a heatmap on a world map to visualize the total CO_2 emissions by country in the year 2023 using the Plotly Express library.

Here's a breakdown:

The multiline comment at the beginning explains the purpose of the code, which is to use the choropleth function from the Plotly Express library to create a map.

fig2 = px.choropleth(...):

This line creates the choropleth map object and assigns it to the variable fig2.

df_2023_only: This is the DataFrame containing the data for the map.

locations="ISO Code": This specifies the column in the DataFrame that contains the location information (ISO codes for countries). locationmode="ISO-3": This tells Plotly that the location codes are in the ISO-3 format.

color="Total CO2 Emissions":

This specifies the column that will be used to determine the color intensity of each country on the map. Countries with higher total CO2 emissions will have a more intense color based on the chosen color scale. color_continuous_scale="Turbo": This sets the color scale to be used for the heatmap. "Turbo" is a specific color scale provided by Plotly. The comment mentions "Viridis" as an alternative. title="Total CO2 Emissions by Country (2023)": This sets the title of the map.

fig2.update_layout(title_x=0.5): This line updates the layout of the map to center the title horizontally.

fig2.show(): This line displays the generated interactive map.

This code is useful for visualizing geographical data, in this case, the distribution of total CO2 emissions across different countries in 2023. The color intensity provides a quick way to see which countries had higher emissions.

End of Activity

Stop and think: What are some things our charts don't show about how these emissions affect people or the environment? How might the data be different if we looked at other years or more countries?”