|  |  |
| --- | --- |
| Title | Hi everyone, and welcome to our presentation on data visualisation  We used this project as an opportunity to explore the different ways we can visualise CO2 emissions, which has been a hot topic lately. |
| **Background and Aims** | |
| Background |  |
| Background |  |
| Aims | Our aims are:  To create an interactive dashboard to visualise CO2 emissions, comparing variables such as population size, GDP, year, and emission type for 205 countries around the world.  To include features and layouts to make it as user friendly as possible. |
| **Quick look at the dashboard** | |
| Dashboard – Global Focus | Here is a quick look at our dashboard  ***(can go onto a web browser and take the class around the dashboard)***  There are 2 sections to it:  Firstly, there is the global section, which places each country on a bubble chart depending on its GDP and Population.  Marker size is dependent on its total emissions, measure in Metric Tons, and markers are coloured according to their continent.  As a point of interactivity, the user can select the year from which to display this data from a dropdown menu. Due to the limitations of the initial dataset we used, we can only look at the years 1970, 1980, 1990, 2000, 2010, and 2020.  Users can also filter by continent, directly on the plotly pot. |
| Dashboard – Global Focus | The global section also contains a bar chart, which displays the “top 10” emissions by country, for a given year and emission type. Again, the interaction is user drive; the user can select the year and emission type from a dropdown menu.  In our Jupyter notebook (not shown on the dashboard), there is a user driven function which takes the year and emission type as provided by the user and prints out a “Top 10” bar chart to the console, as well as exporting it to the output folder, for further future analysis. |
| Dashboard – Country Focus | The second section takes a more country focused look.  Users can select the country and year, and 2 charts will be displayed: a line chart of the Total emissions of that country over the last 100 years, and a pie chart of the emissions breakdown of the selected country for that given year.  An info panel on the top right displays the total emissions for that year.  Again, the plotly chart can be toggled to filter for different emission types.  At the bottom, there is a box containing a link to the global focus, and vice versa. A pulsing effect was added to make it more enticing for the user to click on. |
| **Data Sources** | |
| MetaData Sources - Kaggle | We found 3 datasets on Kaggle:  **CO2 Emissions,** which lists emissions per country, ranging back to as far as 1750. It contains CO2 emissions from various sources for any given year: Coal, Oil, Gas, Flaring, Cement, etc., as well as the Total emissions  **World Population,** which shows the population of each country, every 5-10 years, going back to 1970.  **GDP,** which shows the total GDP per country per year, going back to 1960.  Upon inspection, all 3 data sets had a different number of countries, and the same countries had different name across the data sets.  However, each data set had a “Code” column, which contains a 3 letter code for each country, standardised by the ISO – this is helpful for later during the data cleaning process.  The data sets are downloaded as CSVs, ready to import into our Database. |
| **Data Storage** | |
| Data Storage – MongoDB | The dataset CSVs are then stored in MongoDB as collections under the “Project3” Database, using mongoimport. |
| **Data Cleaning** | |
| Data Extraction | From MongoDB, the datasets are exported into a Jupyter notebook.  The Database, and the collections it contains, are assigned to variables. |
| Data Frames Creation | The next step is data cleaning, in Jupyter Notebook, with Pandas.  All 3 datasets are first converted into Data Frames with Pandas.  Unwanted columns are deleted, and remaining columns are renamed for easier processing further down the line.  The emissions data set contains data ranging back to 1750. As we are only interested in the last 100 years, and as many of the records from the earlier years contain no data, we filtered the data for only years after 1922. |
| Merging | The 3 datasets are merged with Pandas, on the “Code” column as mentioned before.  This was done in order to standardise the country names from all 3 datasets. |
| Data Frame Extraction | The 3 data frames are then “unmerged” – they are extracted from the merged data frame, now with standardised country names.  The data frames are cleaned further by renaming their columns and sorting their records. |
| dtypes | The data types of the cleaned data frames are checked and corrected. |
| Exporting | The cleaned data frames are exported as CSVs, for future projects.  For this project, the cleaned data frames need to be in JSON format to use in our JavaScript code.   * They are first converted to json * The JSON strings are parsed with json.loads * The parsed JSON strings are strung together into 1 JSON output data file and exported into JavaScript |
| Behind the scenes - Jupyter | As mentioned previously, on the side, we added 2 blocks of code to our Jupyter Notebook to increase functionality: one which contains a function to generate a “Top 10” bar chart, and while loop to utilise this function, with user involvement.  Here is the function that we created, which uses the Seaborn library to create a graph, print it to console, and export it to the output folder. |
| Behind the scenes - Jupyter | And here is the while loop, which utilizes the previously established function to keep creating, printing, and exporting graphs for as long as the user wants.  The user just needs to input a year within the timeframe, and an emission type to investigate.  A try/except block is included in case the user enters an invalid year or an erroneous emission type. An error message is printed and the user is prompted to re-enter valid data.  ***(can go into the notebook and show the class how the process in real time)*** |
| **Webpage Development** | |
| Functions | Our JavaScript code initially started off as 1 document, which was then split into 2 files for easier manageability.  We used functions to build charts, initialise the dashboard, and update the dashboard on option change. |
| Loops | Nested for loops were used to population the data arrays which were then used in creating our charts.  Where possible, we aimed to use loops and functions to minimise lines of code.  For example. When creating the traces for the bubble chart, instead of using separate blocks of code for each continent, we wrapped the plot generating code inside a for loop and iterated it through an array of continents. |
| Conditionals | You may have noticed an error message (“Data Unavailable”) on some of the charts in the dashboard.  We coded this with a conditional: the error message displays when no valid option is selected from the dropdown menus. |
| **Webpage Layout** | |
| Webpage Layout - divs | We used divs in our html code to section out or webpage and make it easier for formatting later on.  Click events were used for the dropdown selections. |
| Webpage Styling – CSS | Webpage styling was achieved with an external CSS file. The aim was to create a webpage with a dashboard aesthetic. We briefly had a look on google images to find what features make a good dashboard.  Styling was done concurrently with the webpage design. |
| Webpage Styling – Responsive flex | CSS Flexbox was used to enhance interactivity – flexbox changes the layout and formatting of the webpage dependent on the screen size. E.g. if the webpage is moved to split screen mode., or if used on smaller screens like a phone or a tablet  Config was used in an attempt to make the charts also auto-resize but this only appeared to work for the bubble and line charts. |
| Webpage Styling – CSS keyframes | Keyframes was used to create the pulsing effect on the div containing the internal link, in an attempt to engage the user to want to explore further |
| **Interactivity** | |
| Interactivity | Here we have a listed a few of the elements that we deployed in order to make our website as interactive and user friendly as possible.  Drop down menus are the main feature, while graph toggle and hover are inbuilt functions on plotly.js |
| **Ethics** | |
| Ethical Considerations | We have included an “about” page, which links into the dashboards via a discreet internal link.  This page contains a disclaimer stating that the data that we used is for display purposes only, and credits are given to the original owners, as well as links to Kaggle, the website where we downloaded the datasets from. |
| **Thanks for watching** | |
| Thanks for watching | Thanks for watching!!  Feel free to explore our dashboard here with this link.  Any questions? |