Week 1 Thursday

Data Cleaning and Preprocessing

- Today we'll be working on step 2 of the DSLC which is data cleaning, preprocessing, and exploratory data analysis.
- This stage is the least glamorous and also the one that takes the most time. If the cleaning isn't done properly, your project is basically destroyed.
- Data scientists spend 60% of their time on data cleaning.
- **Data cleaning** is the process of modifying a dataset so that it is tidy, appropriately formatted, and unambiguous
- **Preprocessing** is the process of modifying a dataset so that it satisfies the formatting requirements of a particular analysis or algorithm
- This stage deals with catching errors and missing values
- We have other techniques we can use to transform the data so that we can use it later for algorithms. e.g. standardizing the variables, transforming them to have another distribution, creating new variables/features by combining existing ones
- The dimension is the number of columns in the data (some people include the number of rows). Features, variables, attributes, or covariates are the columns (like a relational database).
- Columns/features can have different data types: numeric, categorical, date and time, short text (structured), long text (unstructured) etc. But all values in a single column must be of the same type.

Table 2.1: US government research and development budget and spending (reported in millions USD). The columns correspond to the year, the budget, the total spending, the spending on climate, the spending on energy, and the political party in office

Year	Budget	Total	Climate	Energy	Party
2000	142,299	1,789,000	2,312	13,350	Democrat
2001	153,197	1,862,800	2,313	14,511	Republican
2002	170,354	2,010,900	2,195	14,718	Republican
2003	192,010	2,159,900	2,689	15,043	Republican
2004	199,104	2,292,800	2,484	15,343	Republican
2005	200,099	2,472,000	2,284	14,717	Republican
2006	199,429	2,655,000	2,004	14,194	Republican
2007	201,827	2,728,700	2,044	14,656	Republican
2008	200,857	2,982,500	2,069	15,298	Republican
2009	201,275	3,517,700	2,346	16,492	Democrat

- Every row is an observation, observational unit, data unit, or data point. So rows are samples and columns are features.

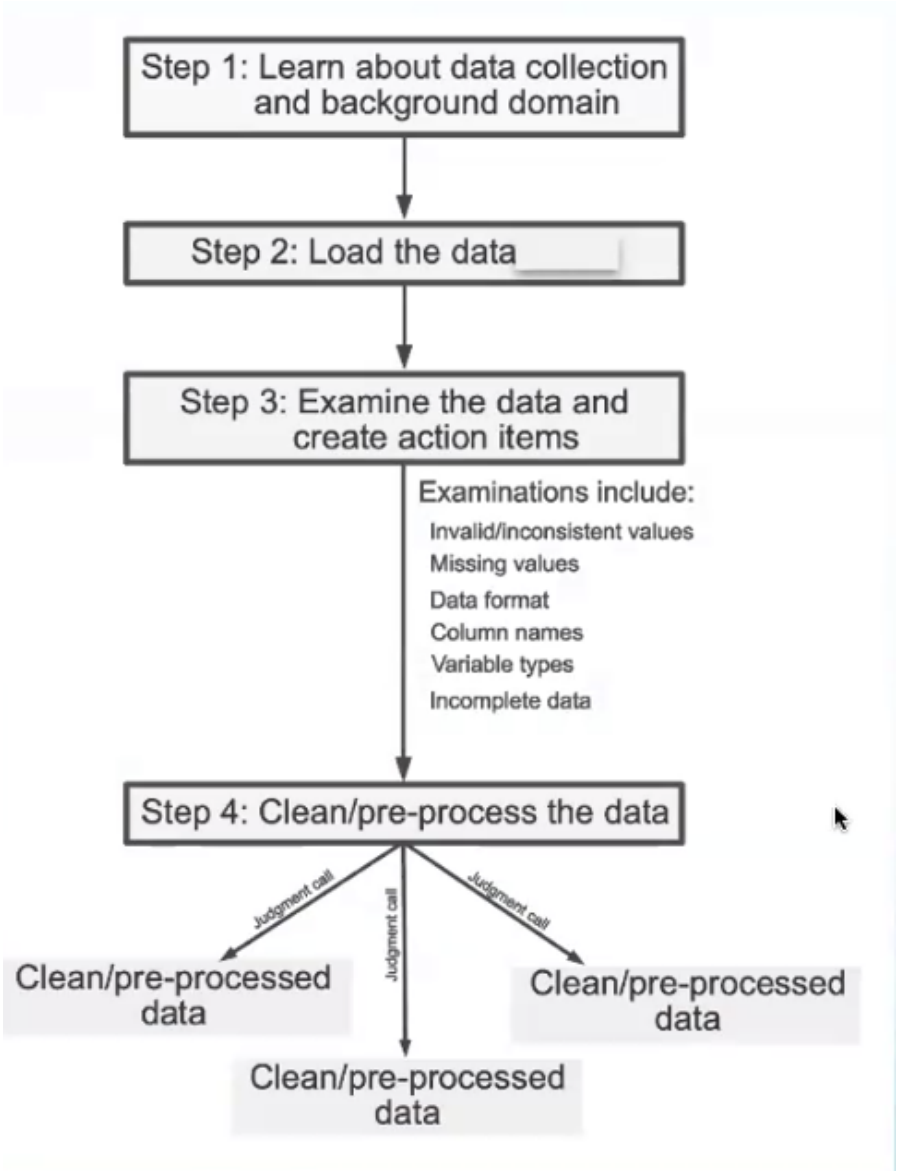
Table 2.1: US government research and development budget and spending (reported in millions USD). The columns correspond to the year, the budget, the total spending, the spending on climate, the spending on energy, and the political party in office

Year	Budget	Total	Climate	Energy	Party
2000	142,299	1,789,000	2,312	13,350	Democrat
2001	153,197	1,862,800	2,313	14,511	Republican
2002	170,354	2,010,900	2,195	14,718	Republican
2003	192,010	2,159,900	2,689	15,043	Republican
2004	199,104	2,292,800	2,484	15,343	Republican
2005	200,099	2,472,000	2,284	14,717	Republican
2006	199,429	2,655,000	2,004	14,194	Republican
2007	201,827	2,728,700	2,044	14,656	Republican
2008	200,857	2,982,500	2,069	15,298	Republican
2009	201,275	3,517,700	2,346	16,492	Democrat

- **Rectangular data** is a type of tabular data. We try to get this data type because it's easy to manipulate and a big part of our data cleaning is ensuring that our data is rectangular.
- There are two kinds of rectangular data: tables and matrices
    - **Tables** (aka DataFrames in R/Python and relations in SQL)
        - Columns have names and different types.

        - Manipulated using data transformation languages (map, filter, group by, join,...) pandas has many features for doing these.
    - **Matrices**
        - Matrices is when we have numeric data, all of the same type (e.g. float, int), and we can manipulate it using linear algebra. numpy library is good for this.
- When would you use one over the other?
    - If you have a wide variety of data types/mixed data types, tables would be better
- CSV are Comma Separated Values. We will use a publicly available survey data from the Global Observatory on Donation and Transplantation. CSV is a very common tabular file format.
    - Records (rows) are delimited by a newline: '\n', "\r\n"
    - Fields (columns) are delimited by commas: ','
    - To read a csv in pandas, use the command `pd.read_csv(header=…)`
    - Each record has each feature separated by commas. End of the record has a newline.
- Each column represents a variable, which is some measurement of a particular concept. It has two common properties:
    - Datatype/Storage Type for a column is the same. Each variable value has the same value stored in memory. But across columns, there can be different data types. With matrices it's different because every single value has to be of the same type.
    - pandas: `df[colname].dtype`
        - Can be integer, floating point, boolean, object (string-like), etc
    - Data type affects which pandas function you can use on a column
- Variable type/feature type: a conceptualized measurement of information (and therefore what values it can take on). It's subject to our interpretation of what that column is.
    - Usually, when understanding the features of a data set, you may not know them a priori so usually you need to use expert knowledge, or explore the data itself. Often not straightforward.
    - Often times, these data sets come with a data cookbook. Consult it if it exists.

    - Affects how you visualize and interpret the data
- There are many data formats. You have to consider whether your data is in a standard format or encoding.
    - **Tabular data**: CSV, TSV, Excel, SQL. You may have access to tabular data from a relational database and use SQL to get access to it.
    - You can have **nested data** formats such as JSON or XML. Nowadays there are python libraries to parse those.
    - Are the data organized in records or nested? If the records are nested, you may actually have to parse data within a record. You may have to think about how you will reasonably unnest the data.
    - Does the data reference other data? Sometimes you may want to start combining different sources of data about the same sample. How do you merge the data? Think about if you really need to. Can we join/merge the data?
    - What are the fields in each record? How are they encoded? (E.g. strings, numbers, binary, dates). It's often counterintuitive. What is the type of the data?
- This is a nice flowchart/checklist to have while you're going through the data cleaning process.



- Step 1: Learn about data collection and domain. First, learn about the data and the background domain knowledge you need. Once you've learned about the data, you want to load the data. That's when the real work happens, because you have to do a type of exploratory data analysis. You have to start looking at the data to see if there are missing values, some values are incorrect, some values don't seem possible, etc. Often times, this is an iterative process: you do the data cleaning once, then you have to go back and do it again based on what you're doing. Finally, you can clean and preprocess the data.
- When you're learning about your dataset, you should ask yourself these questions:

    - What does each variable measure? What real-world quantity is each variable supposed to be capturing? Some of this you can learn from looking at the values, some of it you have to learn from the people who generated the data.
    - How was the data collected? This will affect how the data was measured and how you will interpret it. Mentally visualize the real-world data collection procedure. How was each variable physically measured?
    - What are the observational units? Observational units is a fancy word for data point/sample. Think about what entities each row in the dataset corresponds to.
    - Is the data relevant to my project? What subset of it is relevant? Take a moment to verify that the data that you have helps answer the question you're asking.
    - What questions do I have, and what assumptions am I making? Write down and answer your questions and assumptions.
- Step 2: load the data. In order to load your data, you need to consider how your data is being stored. csv file? txt file? A Microsoft Excel spreadsheet with multiple pages? A database in the cloud? Identify how to load the particular file type into your coding environment (E.g. what function to use).
    - If your data contains multiple tables, is there a key variable that connects them? For more complex data, you may have several tables with the same set of observational units. Identify whether the tables contain any matching (key) variables to merge observational units measurements into a single table.
    - What parts of the data are relevant to the question being asked? Is all the data relevant to your question? If not, consider filtering just to the portion of your data that is relevant to your question. People may have differing opinions on this in a team.
- Often, people get toy datasets that are very clean. But in real data sets, you want to check subsets of the rows, because the first 10 might look great and then when you start randomly sampling, you'll find garbage or incorrect values. Or divvy up the samples among your team. You may have to split up the data cleaning in order to ensure all the bases are covered. Also verify if the dimension is what you expect. Dimension in this context is the number of rows and columns.
- Jupyter notebook: https://colab.research.google.com/drive/1YJvNdyoRp5uVWFjSzyr3nmoNK4RbmXSB?usp=sharing
- First we start off by importing the libraries we need.

```
import pandas as pd
import numpy as np
import plotly.express as px
```

- The observational unit isn't a singular feature. There's not a single ID that identifies the row. Here, the observational unit is the country and the year. We need both of those values to tell if it's a unique sample/data point.
- The data dictionary is found on the website the data is from. It tells you what each column means.
- There are two ways to upload it, using the URL or uploading the file from google drive.
- We want to see that the data we uploaded actually has the columns indicated in the dictionary. To check this, run `organs_original.columns`
- We want to look at the first 20 rows as a sanity check. `organs_original.head(20)`
- When we do that, we can see that the data set has a lot of missing values. Did we load the data improperly, perhaps? Manually inspect the dataset from the original source to check that you didn't load it wrong.
- When we look at the random sampling, we see that some years have missing values, while others don't. This is telling us what something might have happened in the data collection where earlier years didn't have the sample. You can only tell this by talking to a data collector or looking at the data itself.
- Now check the dimension (shape) of the data: `organs_original.shape`. The first number is the number of cols, the next is the number of columns.
- Now we want to do some sanity checks. Is the number of rows divisible by the number of countries? First we need to figure out the number of countries represented in the table. We can do that using nunique: `organs_original["COUNTRY"].nunique()` or `len(organs_original["COUNTRY"].unique()` which gives us 194 unique countries in the dataset. We have 3165 rows. What is going wrong? If we take 3165 and divide it by 194, we don't get an integer. The total number of rows isn't divisible by the number of countries! Also, the description said the data is from 2007-2017, but we saw records from 2000! Also, there's 11 years and 194 countries but we don't have that number of records.
- These are things we have to figure out with more data analysis when we do more exploratory data analysis to help us with data cleaning. In this phase, we are not

actually learning the properties of the data, we're trying to figure out what could be wrong in the data. We have to look to see what could be wrong.

- These are the issues we need to figure out:
  - Why isn't the number of rows divisible by the number of countries?
  - If the study is from 2007-2017, why do we have records from 2000?
- Recap on the global donation data:
  - What does each variable measure? Included in the dictionary in the notebook.
  - How was the data collected? Documented in the link transplant-observatory.org/methodology
  - What are the observational units? For the organ donation data, the donor counts are reported every year for each country, so the observational units are the "country-year" combinations
  - Is the data relevant to my project? Yes, we are seeing how donor rates increase per year per country.
  - What questions do I have, and what assumptions am I making? Some other assumptions we didn't realize we were making: we were assuming that every country would have a measurement for every year of the study, and that wasn't the case. In fact, we had more data points than that, and it wasn't a multiple of the countries. So we either figure out what's going on or decide that we won't deal with these things (which is also an acceptable answer).

Explore Your Data for the Following

- Now we want to figure out what to do when we start seeing problems in our data. Explore your data for the following:
  1. Invalid or inconsistent values: Invalid values are usually impossible measurements. Inconsistent values might be identified when measurements disagree with others in the data set
  2. Improperly formatted missing values: Often when a particular measurement is not available or is not properly entered, it is reported as "missing" (NA) in the data. Missing values should be explicitly formatted as NaN in Python
  3. Nonstandard data format.
  4. Messy column names: Your variable column names should be meaningful and clear to humans.

5. Improper variable types (e.g. everything else is an integer but one is a string). Each variable should have an appropriate type (e.g., numeric, character, logical, date-time, etc.) based on measurement and interpretation
6. Incomplete data. Every single possible observational unit should exist in our dataset for it to be complete. Data for which every observational unit appears exactly once (i.e., none are duplicated and none are missing from the data) is considered complete.

- What is wrong with the following data?

#	Country	Alcohol	Deaths	Heart	Liver
1	Australia	2.5	785	211	15.30000019
2	Austria	3.000000095	863	167	45.59999847
3	Belg/Lux	2.900000095	883	131	20.70000076
4	Canada	2.400000095	793	NA	16.39999962
5	Denmark	2.900000095	971	220	23.89999962
6	Finland	0.800000012	970	297	19
7	France	9.100000381	751	11	37.90000153
8	Iceland	-0.800000012	743	211	11.19999981
9	Ireland	0.699999988	1000	300	6.5
10	Israel	0.600000024	-834	183	13.69999981
11	Italy	27.900000095	775	107	42.20000076
12	Japan	1.5	680	36	23.20000076
13	Netherlands	1.799999952	773	167	9.199999809
14	New Zealand	1.899999976	916	266	7.699999809
15	Norway	0.0800000012	806	227	12.19999981
16	Spain	6.5	724	NA	NA
17	Sweden	1.600000024	743	207	11.19999981
18	Switzerland	5.800000191	693	115	20.29999924
19	UK	1.299999952	941	285	10.30000019
20	US	1.200000048	926	199	22.10000038
21	West Germany	2.700000048	861	172	36.70000076

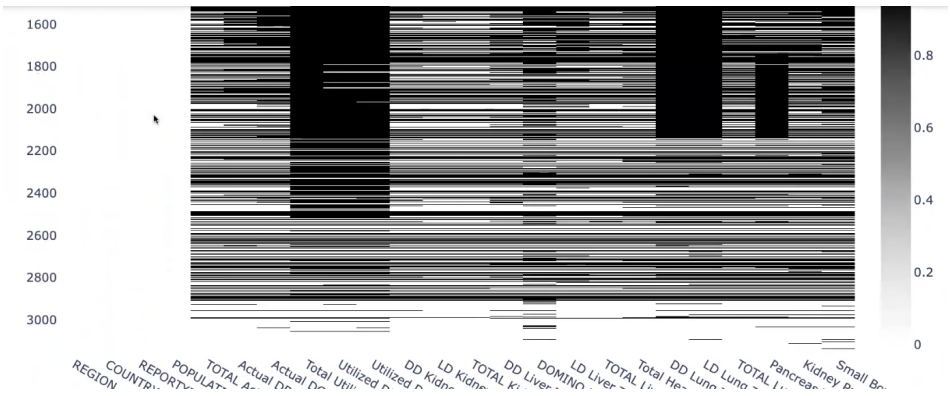
- There are some negative numbers. People can't drink a negative amount of alcohol.
- Alcohol consumption is liters per capita of alcohol consumption. The deaths is total number of deaths from alcohol per 1000 people. Heart is the number of deaths per 100,000 from heart disease. Liver is the number of deaths per 100,000 from liver disease. Just looking at this chart, you have no idea what the columns meant.
- Missing values. In Spain, we don't know the rate of heart disease or liver disease deaths. What can we do with those? That's a judgement call. You could take the average value and substitute that in. Or we could drop the record from the study.
- To check for invalid or inconsistent values:

1. Inspect random subsets of rows of the data
2. Check ranges of values for numeric data. e.g. negative values aren't correct for alcohol consumption
3. Create histograms for numeric data. This will tell us if we have any outliers.
4. Check unique values of categorical variables to ensure they make sense.

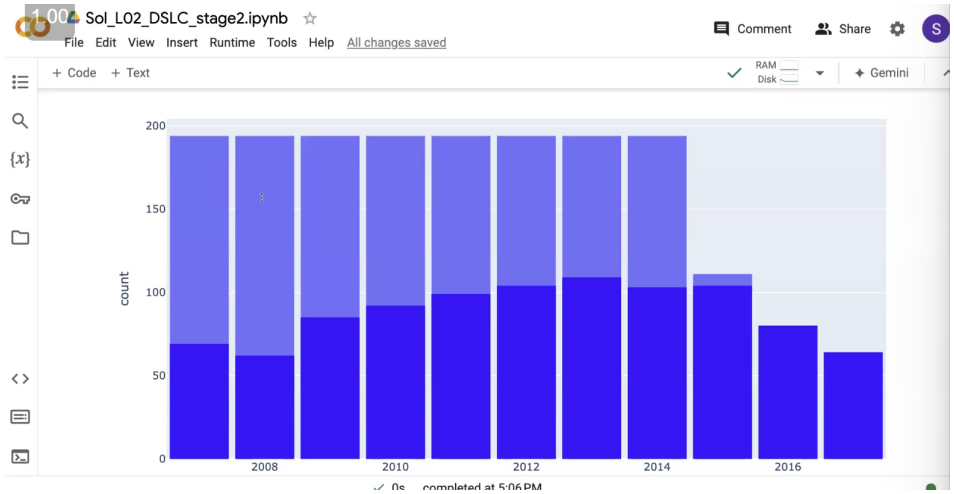
- Action items:
  1. Leave it alone
  2. Substitute a valid correct value
  3. Replace with NaN
  4. Convert values to some common unit
- These are judgement calls you have to make as a team.
- To identify missing values:
  1. Check ranges of values for numeric data
  2. Create histograms for numeric data
  3. Check unique values of categorical variables
  4. Calculate the proportions of missing values. Why is this important? You want to see what you can actually trust from that column. e.g. if you have 75% missing values, that feature may not be something you should use in your study.
  5. Visualize the pattern of missing values as with a heatmap. This tells you if there might have been something with the data collection that you didn't know about that you should know about. e.g. when we opened up the CSV, almost all countries had no values for the year 2000, which makes us think that a lot of countries maybe didn't respond to the survey, so we shouldn't include those years. There's no way to know this without looking.
- Action items for missing values:
  - Make sure they're marked as NA
  - Replace known values and justify your choice.
  - Apply imputation. There are many methods for imputation such as:
    - **constant-value imputation**: replacing all missing values with a plausible constant value

        - **Mean imputation**: replacing all missing values in a column with the average value of the non-missing values in the column
        - **forward/backward fill imputation**: in time-dependent data, replace with previous or next data point
    - Set a threshold for the percentage of missing values and drop all features that meet that.
    - Remove rows with missing values. You have to document and justify the decision. e.g. for the organs dataset, we might not want to include data from the year 2000 because there's so much missing data.
- Step 3: examine data and create action items. Check that the dataset is **tidy**. What is a tidy dataset? A tidy dataset satisfies the following criteria:
    - each row corresponds to a single observational unit (e.g. combination of country and year).
    - Each column corresponds to one single data type of measurement
- Other things we can do is to check variable names and rename them to more meaningful names. A human without access to column dictionary should be able to read the column names and understand what they are.
- Check the type of variables in each column. Make sure that the type matches the variable and the values match the column.
- Check if your dataset is **complete**: a complete dataset is one in which each observational unit is *explicitly* represented in the data.
- Action items for complete data set:
    - choose a subset that is complete. Document and justify this choice.
    - Add the missing observational units to data and populate them as you would missing values. Imputation.
- While you're cleaning the data, make sure you save the raw data!!
    1. Recommendation: writing a data cleaning function whose input is the raw data and whose output is the clean data
    2. In this case it would not be necessary to store cleaned data, but may be preferable for your project if data cleaning is time and computationally intensive.
- There's an example of an imputation function in the notebook. We impute the data with the average of the value before and after it.

- It can be useful to use a heat map to visualize the missing values and which fields were actually having the missing values.



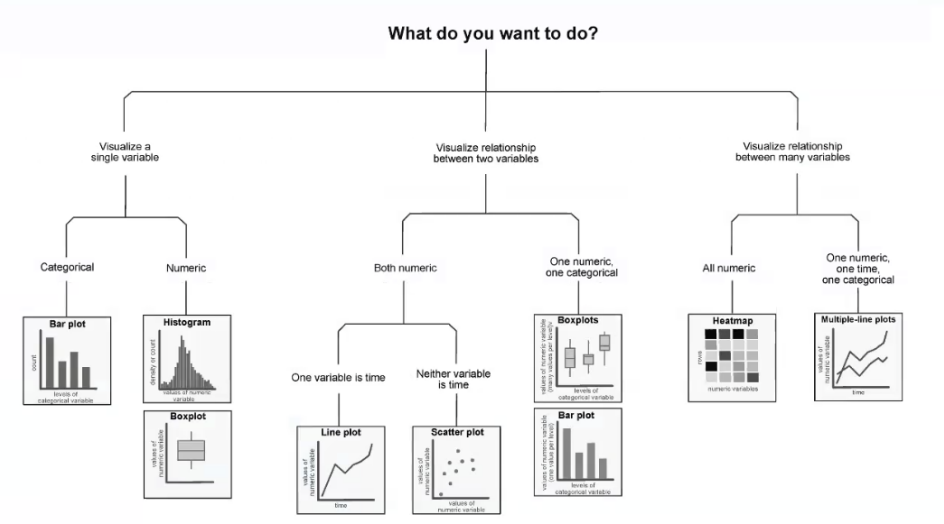
- Heatmap can be used both for data cleaning and exploratory data analysis. Sometimes you just want to get a quick look of which has the greatest percentage of missing data. We calculated the percentage of missing values for each variable. Once we calculate the proportion, we can plot as a heatmap where the darker color means more missing. Instead of looking through all your data, you can plot it as a heat map to identify features you need to spend more time looking at.
- Then to actually deal with what you'll do with the missing data, that's your judgement call. The notebook has an example of how you can use the average to fill in the missing values.
- We can also change the column names to be more understandable.
- You can check the missingness, counts for when it was high vs. low, etc. You can start overlaying the bar plots on each other so that you don't have to manually go through each record. You can start getting summaries to get a sense of when problems might arise.



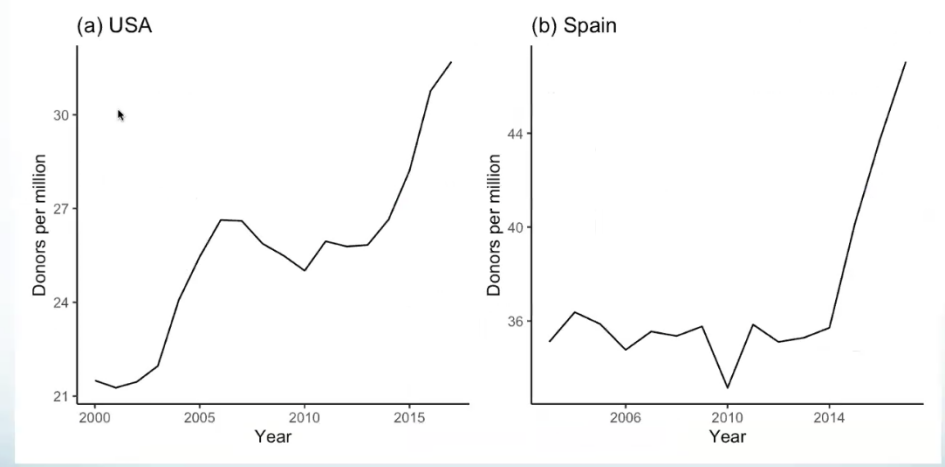
- The function renames all the columns so that the column names are meaningful to humans.
- The other benefit of writing a cleaning function is reproducibility, especially if you document the choices you make.
- Once you have the clean dataframe, then you start checking the rows again and running the sanity checks again. Essentially, you're treating the cleaned dataframe as though it's a new dataset.
- Another thing you can do is in your new dataset, you keep a column for the original and a column for the imputed, just to see the effect it has on your analysis. When rerunning the sanity checks and properties, you can compare the results from the original vs. cleaned values.

Exploratory Data Analysis

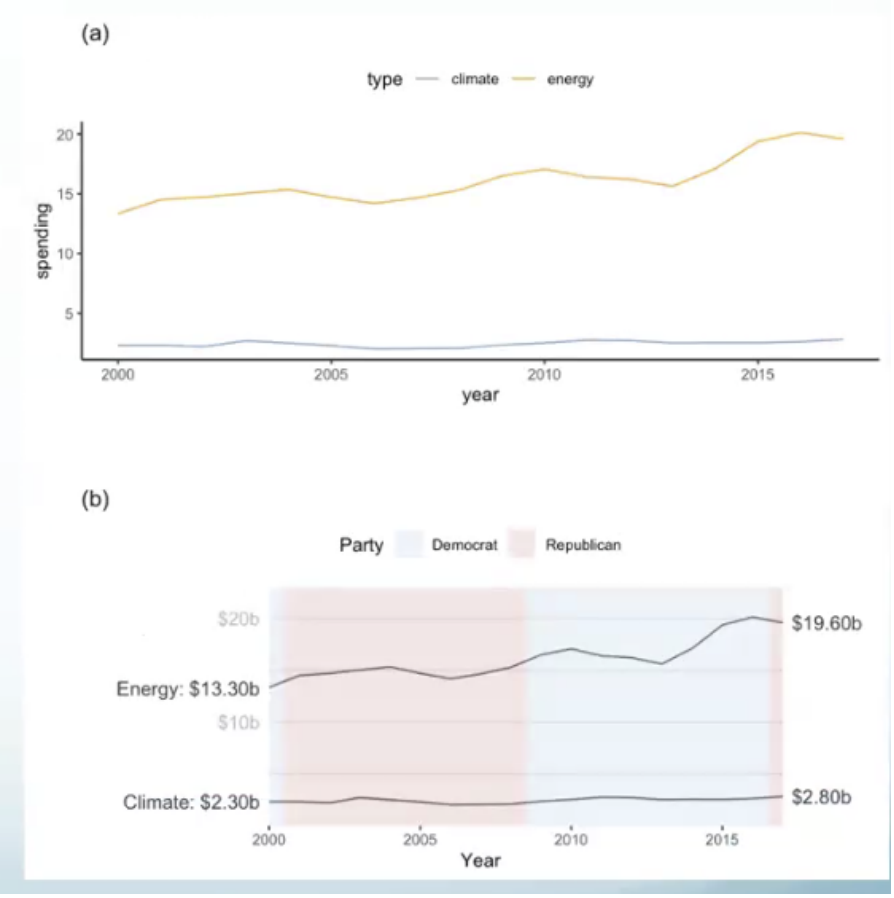
- **Exploratory Data Analysis (EDA)** is the task of visually and numerically summarizing the patterns, trends, and relationships that a dataset contains in the context of a domain problem. So we are going to see the relationships between variables now. Whereas before we were just exploring the data to see if there were problems, now we want to explore whether there are meaningful relationships between the variables in the data that we can exploit or study further.
- John Tukey: "Exploratory data analysis is an attitude, a state of flexibility, a willingness to look for those things that we believe are not there, as well as those that we believe to be there."
- Here's a flow chart we can use for doing exploratory data analysis:



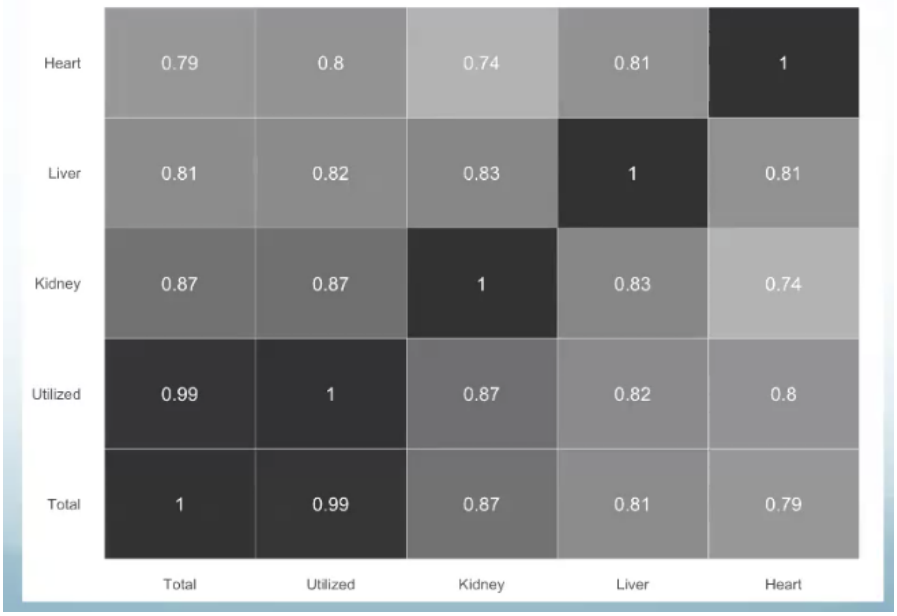
- e.g. if you want to visualize a single categorical variable, you can just use a barplot. If it's numeric, we may want to use a histogram or boxplot.
- If you want to visualize the relationship between two variables, if they're both numeric and one is time, you can have a line plot where the x-axis is time and the y-axis is the variable we're measuring. If neither variable is time, you can just do scatterplots. If we have one numeric variable and the other categorical, we can make boxplots where the categorical variables are on the x-axis and the numeric variables are the y-axis.
- If we want to visualize many relationships that are all numeric, we can make a heatmap.
- If you have one numeric, one time, and one categorical, you can make many lineplots on top of each other, but the challenge with that is doing so in a way where you can get something meaningful out of it. e.g. you might want to pick out and highlight some lines because you suspect that those might be the countries with the highest donor rates. It's tricky but helps you summarize the data quickly and identify trends to highlight.
- Also make sure that the variables are comparable (e.g. on similar scales). e.g. what could go wrong here? The x and y axis scales are different. The image is misleading because it makes it seem as if Spain has fewer donors, but the y axis for Spain starts off after the y axis for USA ends, so in reality Spain actually has more donors.



- Difference between exploratory and explanatory data analysis:
  - Exploratory Data analysis (EDA) creates numeric and visual summaries of the data for understanding patterns in it
  - Explanatory data analysis polishes the most informative exploratory tables and graphs to communicate them to external audiences



- E.g. for the organ donations example, you can take the multiline plot and plot Spain's data in a different color and the USA's data in a different color. That's one way to make explanatory diagrams.
- Nonvisual EDA:
    1. Calculating summary statistics such as mean, median, variance, and standard deviations of variables
    2. Caution: histogram of variable reveals more information. The summary statistics don't tell you much about the distribution. There are many distributions with the same first and second moment. So doing a histogram can be beneficial instead of only computing those summary statistics.
    3. Calculate the covariance and correlation to check for linear relationships. You may just want to do a sanity check to check for correlations among variables.
- Handling correlations in EDA:
    1. Linear relationships can also be checked using scatterplots in addition to correlation. If you don't just want to rely on the correlations, you can make scatterplots to get an idea of the relationship or correlation between variables.
    2. You can create a heatmap which is the correlation between every pair of variables in the data. This is very valuable because if you get a dataset for your project and you don't know yet what you want to look at, if you look at the features that are highly correlated, that can give you ideas of what to explore further. Something like this is a quick way to check if variables may be related in linear relationships.



- Question: how do the donation rates change over time for each country? To explore this, we will use:
  - line plots
  - scatterplots
  - correlations
  - bar graphs
  - heatmaps