**Venezuela Economic Downfall**

Project 2 Group 7: Laura, Jen, Jarod, Virgil and Jill

Project

Our group chose this project because we wanted to see the effects of crisis on a country.

Sources of Data:

1. Obtained news data from New York Times and The Guardian.
2. Obtained a gross national income per capita csv data file from <https://data.humdata.org/dataset/who-data-for-venezuela-bolivarian-republic-of>.
3. Obtained Exchange rate of VEF/USD for 2000-2019 from <https://www.federalreserve.gov/releases/h10/current/>.
4. Obtained a demographic data csv file from World Bank <https://data.worldbank.org/country/venezuela-rb>.
5. Obtained economy data from The Index of Economic Freedom <https://www.heritage.org/index/country/venezuela>
6. Obtained additional news data from <http://news-api.org/>

Steps Required to Reproduce ETL:

1. Extract
   1. New York Times.
      1. Used python to access NY Times API, extracting news snippets, lead paragraph, URL, headline, keywords (list), and publication date for 01/01/2014 through 01/01/2019, using requests.
      2. Used a loop to page through the query results (10 per page) and time.sleep(6) to avoid call limit.
      3. Put each dictionary of info into a main list.
      4. Note: NYT API instructions say to use &offset=x to paginate, but that does not work. &page=x is the right nomenclature. Offset kept returning the same 10 articles.
      5. Loaded the list of dictionaries into a local MongoDB instance to the collection nyt\_news using pymongo in Jupyter Notebooks.
      6. Pulled the URL for each document from the collection into a list. Then iterated through that list, scraping the NYT website using BeautifulSoup (‘div’, class\_='css-53u6y8'). The paywall only allows scraping of the first two paragraphs. The Guardian ('div', class\_="content\_\_article-body") allowed scraping of the full article text.
      7. Updated the MongoDB entry with the article text (HTML) by matching the URL.
      8. Note: Tried to use the parameter verify=False in the requests.get statement to get around the NYT https verification, but that didn’t work.
      9. 532 articles from the New York Times.
   2. The Guardian
      1. Used a very similar approach with The Guardian’s API. However, their limits were more generous.
      2. Used page\_size=100 parameter and looped through 20 times to get all the articles, using an if statement articles['response']['status']==’ok’ to keep going until we reached the end of the list, where response status was an error.
      3. Extracted 581 articles from The Guardian.
      4. The news API calls and scrapes were then made into python functions, used PyMongo to test directly inserting into the cloud, which after moving off the Mac on Mojave and onto High Sierra, worked just fine.
   3. Gross National Income Per Capita was a CSV file we obtained from <https://data.humdata.org/dataset/who-data-for-venezuela-bolivarian-republic-of>.
   4. Federal reserve
      1. Navigated to country data page and downloaded a csv containing monthly exchange average for 2000-2019.
   5. World Bank
      1. Navigated to Venezuela page and downloaded a csv file containing 1500 rows of demographic data for 1960-2018.
   6. The Index of Economic Freedom
      1. Explored the economic data and filtered the data specific to the country of Venezuela for 2013-2019 and downloaded the csv file.
   7. Additional news
      1. Used an API to scrape news headlines and snippets from a variety of sources

1. Transform
   1. News:
      1. When tried to transform, noted an issue which required us to go back and re-extract.
         1. NYT data was tricky because the API documentation said to use ‘offset’ to paginate - returning a repeated set of the same articles. Using ‘page’ instead led to a unique dataset of articles of varying relatedness to Venezuela.
      2. Guardian data was cleaned by removing articles of type liveblog after insertion into Mongodb, and duplicates were removed by eliminating records with no article text as the update adding the text only worked on the first document matching the URL.
      3. In retrospect, collecting API URL from the Guardian was a mistake, and just url, headline, date and source were enough before scraping to get the copy.
   2. Gross National Income Per Capita.
      1. We imported the csv file into a jupyter notebook and used pandas to clean the table. Specifically, grabbed the specific columns needed, renamed the columns, formatted the amount, set the index, and ordered dataframe by year.
   3. Exchange rate data did not need cleaning.
   4. World Bank.
      1. We imported the csv file into a jupyter notebook and used pandas to clean the table. Specifically, grabbed the specific columns needed, renamed the columns, dropped rows with null values, formatted the amount, set the index, and ordered dataframe by indicator name.
   5. The Index of Economic Freedom
      1. We imported the csv file into jupyter notebook and use pandas to convert to a dataframe. Then transform it to dictionary format and load to MongoDB.
   6. Additional news data did not need cleaning.
2. Load
   1. The final production is non-relational database (heroku\_pptb0bm8) loaded to the cloud
   2. The final collections used in the production database are:
      1. news\_import
      2. news
      3. VEF\_Exchange\_Rate\_data
      4. gross\_nat\_inc\_per\_capita
      5. worldbank
      6. measurements
      7. news\_api