Project Report Gavin Horan

# GitHub URL

https://github.com/gwavin/UCDPA\_gavinhoran.git

# Abstract

I set out to focus on COVID data. After searching several resources, I settled on the European Centre for Disease Control (ECDC) COVID Data; there are a range of datasets, these cover all aspects of COVID pandemic data as captured by the ECDC and hosted at <https://www.ecdc.europa.eu/en/covid-19/data>. I used a variety of means to download and inspect the data for insight.

# Introduction

The SARS-COV-2 pandemic is the most significant health emergency in modern history. The pandemic lifecycle may be the same as those of the past, but now we can record vastly more data than previously possible. It is difficult to make sense from this torrent of information.

Hospital facilities may be affected by COVID; for example we may have to make a decision as to whether or not we close or limit access to the pharmacy.

Weekly analysis of the datasets collated by the ECDC may give us some insight which would assist our decision-making.

I decided to focus mainly on Ireland, and to an extent on those countries I considered to have health systems and populations closely matched to our own; the Netherlands, Germany, Spain and France.

# Dataset

# The ECDC is an EU agency which engages in disease surveillance, epidemic intelligence, and response, among a wide range of other roles.

# As a result of this, they have an excellent selection of datasets from which I drew to build reports which I hoped would be informative. These were located at <https://www.ecdc.europa.eu/en/covid-19/data> and it was here that I really found the most interesting data to manipulate.

# I did not start with the ECDC; Kaggle is an invaluable resource for all manner of datasets and this is where I first looked. From here I took the fetal\_health.csv. I also downloaded NHANESdata233052221221.csv which is the data from the CDC National Health And Nutrition Examination Survey.

# Implementation Process

# In order to harness the power of API’s, I sampled some datasets from Kaggle.com.

# The Kaggle command line interface (CLI) tool is the easiest way to interact with Kaggle’s public API.

# “*pip install Kaggle*” from the terminal, ran in PyCharm. This used the python package manager, PIP, to install Kaggle.

# Instructions on the use of Kaggle are saved in Kaggle.txt. This is the Kaggle help page.

# *kaggle datasets list -s health > KaggleHealthList3.csv* downloaded the list of Health data files which were available for examination and saved these at KaggleHealthList2.csv.

# *kaggle datasets download andrewmvd/fetal-health-classification* saved the files placed on Kaggle by the user andrewmvd. I hoped that these might have information which would be of interest, so I spent some time working on those.

# In my first program, *fetalHealthchecker.py* I imported the fetal\_health.csv CSV file into a Pandas DataFrame, to see if it would be of any interest. I printed the .info() and the .head() and unfortunately, didn’t really find much of value to look at. I was sure Kaggle is full of great information. I examined the NHANES dataset, in a similar fashion. This was at <https://healthdata.gov/w/wwc6-um3r/default?cur=kq4-OidPS0Z> They provide a similar tool, in the nhanes package, which allows the downloading and inspection of their data. However, again, I had no interest in this. The very cursory file, usingNHANES.py has the initial investigation included. NHANESdata233052221221.csv was the output from this; I used a similar CLI tool to take in the dataframe, and used a function which I wrote, and placed in functionFile.py, to use pd.read\_csv() to send it to a named file.

# At this point, I cast the net wider and found the ECDC website.

# I actually downloaded the .csv using the mouse, which is not good enough.

# I used a web scraping method in *myExampleOfScraping.py* to demonstrate that I can actually use the methods of webscraping that we learned in the course.

# The comments at the top of this program indicate the steps I took in this program. I used Beautiful Soup to handle the html and parse it into a dataframe.

The *BeautifulSoup.soup.findall*() pulls all html links out of this; These are then inserted into the *links* list. I used a list because it was not important to me that there be a key available to access the items, as in a dictionary. I wanted to run through all the values in links and inspect them with a regex.

items in dictionaries are accessed via keys and not via their position; this was not necessary.

Lists can have a variety of types and there have a problem with this; I couldn't run a regex where the system may have encountered ints and floats.

I converted all entries in *links* to strings, using the built-in *.str()* function, and then ran a regex to find those ending in .csv. I knew there would be at least one. There may

be more in future; that is the joy of this, I can run this program again to find or indeed "scrape" as many .csv

files as a page holds.

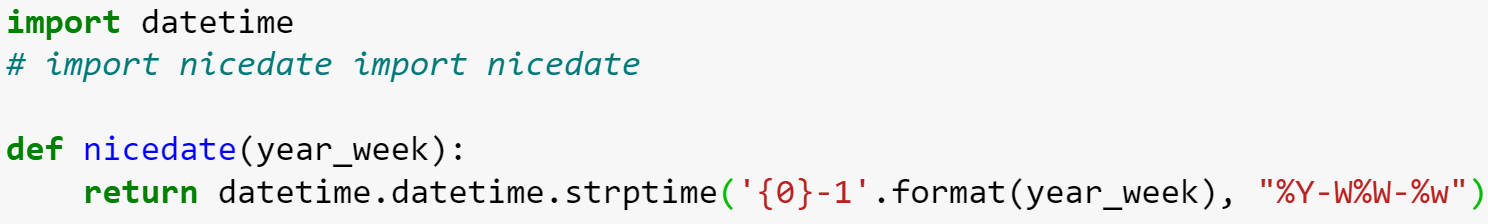
Next I converted the list to a dataframe, because I find it easier to manipulate these.

I then used a function I had already written to push this out to a .csv file. This includes the date and time

appended at the end using a function which I prepared already.

I created a function for re-use with the ECDC dataset; they have an annoying habit of using 2020-W30 etc to refer to the 30th week of the year.

I rewrote a code snippet to deal with this;



This takes in the nasty format, interprets it by slicing out the year, and the week number, and then incorporating those into a datetime which includes day 1 by default. This meant that the labels at the bottom of my graph could look much nicer.

In order to become familiar with the various utilities that one might take advantage of I also created and used a Jupyter notebook for the project.

# Results

I got some nice charts;

Put these below; everything essentially that is a .py in my UCDPA folder on github.

On the evening of the 23rd

# Insights

I'm going with "This regression analysis shows that the impact of the independent variable is not constant. Early in the pandemic, tests done were linked strongly to new cases diagnosed. This implies that the increase in testing facilities early in the pandemic meant that we were catching a lot but not all of the cases that we might. However, since we have increased testing capacity, peaks are more easily detectable, since when a "wave" of infection begins, or ends, this is visible in the test results returned. As the number of available testing facilities has grown to an appropriate level, tests.

Holland seem to be going slightly ahead of us.

Lots of peaks indicating that there was a wave; these speak of

# References

(Include any references if required)