Michael Madigan

mikelispopple@gmail.com

A picture containing clipart

Description automatically generated

Data analysis of the economic and societal impact of covid 19

Project Report

Table of Contents

[Abstract 1](#_Toc69632196)

[Introduction 2](#_Toc69632197)

[Datasets 2](#_Toc69632198)

[Implementation Process 3](#_Toc69632199)

[First data set 5](#_Toc69632200)

[Second data set 6](#_Toc69632201)

[Thirds data set 7](#_Toc69632202)

[Merging 8](#_Toc69632203)

[Merge-1 8](#_Toc69632204)

[Merge-2 8](#_Toc69632205)

[Plotting 9](#_Toc69632206)

[Results 9](#_Toc69632207)

[Insights 9](#_Toc69632208)

[References 11](#_Toc69632209)

[Bibliography 11](#_Toc69632210)

**GitHub** URL <https://github.com/MM2020-11/UCDPA_Michael_Madigan>

# Abstract

The arrival of Covid 19 virus has impacted Ireland’s society and economy. It has invaded our daily lives and is constantly in the airways, news and internet. This report applies data analytics to visualise the societal and economic impacts. Link ref 1

The python programming language is used to process, clean and display data graphically allowing data driven conclusions. The report uses three main data sources; covid new daily cases, staying local data and weekly pandemic support payments. These data sources are compared and analysed.

This report delves into Irish covid reports and completes the following:

1. Presents and analyses Irish covid cases data.
2. Shows impact on government support payments.
3. Analyses data from ‘staying local’ metrics collected from mobile phone GPS location data
4. Checks for correlation between trends
5. Checks for disparities between male/female payments

# Introduction

As covid is so prevalent and wide spread a vast amount of data is collected world-wide. This large data set allows comparison and contrast between different data sets using the Python programming language.

Using learnings from the UCD PA course and on-line training allowed processing of data using python tools. Python has an extensive library of add-on features which makes data manipulation all the easirer. Also python libraries include plot tools matplotlib and Seaborn. This project uses the Matplotlib library.

# Datasets

The project used three separate data sources.

The first is a direct link to covid world data via URL link to Python data frame. This data source is very comprehensive and updated daily, hence a very useful data set. This data contains world wide data and includes Irish covid case numbers. This data is collated on a daily bases at a web site “our world in data” [1]

The second data source is from the Irish government site www.CSO.ie and called the “Staying Local Indicator”. This data was collected from a mobile phone company "Three Ireland" and analysed. Anonymised data is calculated for people staying with 10km of the home based on phone GPS data. It is believed that increased people movement should correlate strongly to increased transmission of covid. This data is collected at [Link](#https://ws.cso.ie/public/api.jsonrpc?data=%7B%22jsonrpc%22:%222.0%22,%22method%22:%22PxStat.Data.Cube_API.ReadDataset%22,%22params%22:%7B%22class%22:%22query%22,%22id%22:%5B%5D,%22dimension%22:%7B%7D,%22extension%22:%7B%22pivot%22:null,%22codes%22:false,%) and [2]. This data is collected daily by county and state

A third data set came from Irish government records of Pandemic Unemployment Payments, this data is collected weekly and catagories by age and sex. In order to align the data sets this data was interpolated into daily records to match with the two previous data sets and in order to make direct comparisons. Details at [3], Data at [4]

# Implementation Process

High level graphical overview

Pandemic Payments data

Staying local data

Covid data

Read

data

from local CSV file

Read

data

from local CSV file

Download data

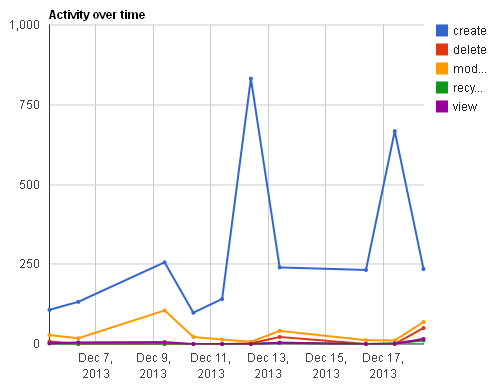
Direct from url

Correct date data

Interpolate missing daily data

Merged

Merged

Graphical Display

Graphical Display

Graphical Display

Graphical Display

Data cleaned, check, sliced and diced

Data cleaned, check, sliced and diced

Data cleaned, check, sliced and diced

Implementation

This project was implemented using the Pycharm IDE and python. The program is contained in one file “main.py”.

A high level process graphically on previous page. The project reads files both local and URL link. Clean, modify and merge data sets. The called functions as

1. Pandas as pd
   1. used to efficiently manipulate the datasets
2. Matplotlib.pyplot as plt
   1. Graphically represent data
   2. Selected the OO style for plotting controls
3. numpy as np
   1. For numerical calculations
   2. This function was only used during development phase
4. datetime as dt
   1. Used to process and clean date/time formats

To help in the debugging process the code save data frames to CSV files for checking and uses print commands to check calculation and data properties. The debugging CSV files are named “export\_xxx” for consistency. Print commands include line 44 or parameter name.

## First data set

The first data source contains covid pandemic data from our World in data and a pandas connect directly to the website containing the file URL ‘owid-covid-data.csv'. The file was inspected with .head and .tail, .shape and .dttypes to determine the dataframe design.

This data file contained a long list of columns. To review the list a python “FOR” loop was used to print the column list ( as an neater alternative to using the .columns property) list using

for idx, column in enumerate(df1.columns):

print(idx, column)

On reviewing the resulting columns were selected and just the Irish iso\_code was extracted for country to allow select of IRL for Ireland, date and number of new cases. In both instances “.loc” was used.

Selected\_cols = ['iso\_code', 'date', 'new\_cases']

The .loc pandas function was used to separate the selected columns

df\_iso\_date\_new\_cases = df1.loc[:, selected\_cols]

Once the data frame was created to had to be cleaned as follows; Nan validation and fill with zeros. The data was cleaned to removed missing points which read as NaN and replace newcases with zeros

df.fillna(0)

The dataframe was tested for NaNs using a isna() and sum() Pandas function, then printing the result.  
count\_NaN = df\_iso\_date\_new\_cases\_no\_NaN[selected\_cols].isna().sum()

On review the data types found date was not in a date data type. pd.to\_datetime function was used to convert to date

pd.to\_datetime(df\_iso\_date\_new\_cases['date'], yearfirst=True, format="%d/%m/%Y")

The data was reduced to select only data rows relevant to Ireland

df\_IRL = df\_iso\_date\_new\_cases\_no\_NaN.loc[df\_iso\_date\_new\_cases\_no\_NaN["iso\_code"] == 'IRL', :]

During debugging I found that the on-line data was not always available, A local version is included but will not be as up to date as the on-line version

## Second data set

Second dataset from [www.cso.ie](http://www.cso.ie) was the Staying local indicator data which was manually downloaded as a CSV file and read into the Python code.

A pandas command was used to read from a local downloaded CSV file. Data was inspected, checked for NaN’s

Data was inspected for columns using a “for loop” and “.head()” functions as small number of columns.

The required columns were extracted using “.loc”

required\_cols = ['Date', 'County', 'VALUE']

df\_SLI\_Date\_County\_Value = df\_SLI.loc[:, required\_cols]

For unknown reason the data contained an incorrectly formatted date on 20 December21, missing a space. This was fixed with code to replace the error as this alarmed when completing the date data type conversion.

df\_SLI\_Date\_County\_Value['Date'] = df\_SLI\_Date\_County\_Value['Date'].str.replace('2020 December20', '2020 December 20')

This data has 4 incorrect missing data point so the graph dropped to zero at this data points see Figure 1. below To improve the presentation the data was interpolated using pandas .interpolate funcition.

df\_pup2 = df\_cases\_pup1.interpolate(axis=0)

This removed this zero points shown in .Figure 1 for a smooth graph as shown in Figure 1

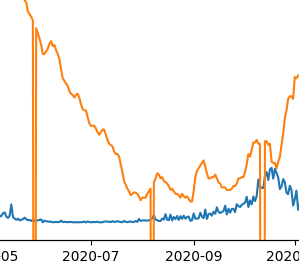
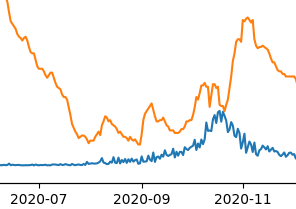
 

Figure 1 graph before interpolation Figure 2 graph after interpolation

A subset by ‘state’ using the .loc function was used to create a new data frame. The comparison in country wide at ‘state’ and not a specific county level.

df\_SLI\_state = df\_SLI\_Date\_County\_Value\_No\_NaN.loc[df\_SLI\_Date\_County\_Value\_No\_NaN["County"] == 'State', :]

## Third data set

Data was download from CSO.ie containing the Irish pandemic payments. As per previous data set the data was read in from a local CSV file, cleaned off NaN’s This files contained dates in the format as weeks in format YYYYWXX eg 2020W12. Code was required to convert this format to a date format.

The code added an extra day format for the first Monday of the week

df\_pup['first\_Monday'] = df\_pup['Week'] + '-1'

Then date formatted with to\_date function

df\_pup['first\_Monday\_date'] = pd.to\_datetime(df\_pup['first\_Monday'], yearfirst=True, format="%YW%W-%w")

The required columns were extracted

required\_cols = ['Statistic', 'Age Group', 'Sex', 'VALUE', 'first\_Monday\_date']

df\_pup1 = df\_pup.loc[:, required\_cols]

then the relevant rows were extacted in a multi-step process

df\_pup1 = df\_pup1.loc[df\_pup1["Statistic"] == 'Persons in receipt of the Pandemic Unemployment Payment', :]

df\_pup1 = df\_pup1.loc[df\_pup1["Age Group"] == 'All ages', :]

df\_pup1 = df\_pup1.loc[df\_pup1["Sex"] == 'Both sexes', :]

Second steps

required\_cols = ['Statistic', 'VALUE', 'first\_Monday\_date']

df\_pup1 = df\_pup1.loc[:, required\_cols]

## Merging

### Merge-1

Date from covid cases and staying local were merged

Data was merged using a left join matching the two data frames dates as indices

df\_cases\_County = pd.merge(df\_IRL, df\_SLI\_state, left\_on='date', right\_on='New\_Date', how='left')

The merged data frame was further reduced and cleaned

df\_cases\_County\_cleaned = (df\_cases\_County\_2[(df\_cases\_County\_2[date\_cols] != 0).all(axis=1)])

After merging and graphing, a calculation to check for correlation was performed to compare the pup receipients to cases

correlation = df\_cases\_County\_2['VALUE'].corr( df\_cases\_County\_2['new\_cases'], method='Pearson')

print('\n coor1 = ', correlation)

The Pearson’s coefficient data is discussed in results section

### Merge-2

Data from Irish covid cases and pandemic payments were merged

Data was merged using a left join matching the two data frames dates as indices

df\_cases\_pup1 = pd.merge(df\_IRL, df\_pup1, left\_on='date', right\_on='first\_Monday\_date', how='left')

As the pup data was in a weekly format is was necessary to interpolate to report on a daily bases in line with the previous two data sets

df\_pup2 = df\_cases\_pup1.interpolate(axis=0)

Again this data was tested for correlation against the cases data

correlation = df\_pup2['VALUE'].corr( df\_pup2['new\_cases'],method='Pearson')

print('\n coor2 = ', correlation )

The Pearson’s coefficient data is discussed in results section

### Plotting

Data was plotted and displayed in the reports section. This used matplotlib for plots and bar graphs. Also annotations were added to expand the information. Dual axis were used with limits when the data ranges were different. The recommended object-oriented approach was used

The data was extracted from the data frames and converted to lists. Whilst there are many methods to plot data this solution was found to be convenient by not memory efficient.

Titles, axis labels are included in this code.

The data was arranged using ‘grouped by’ features in preparation for plotting on bar charts

# 

# Results

<<Include the charts and describe them>>

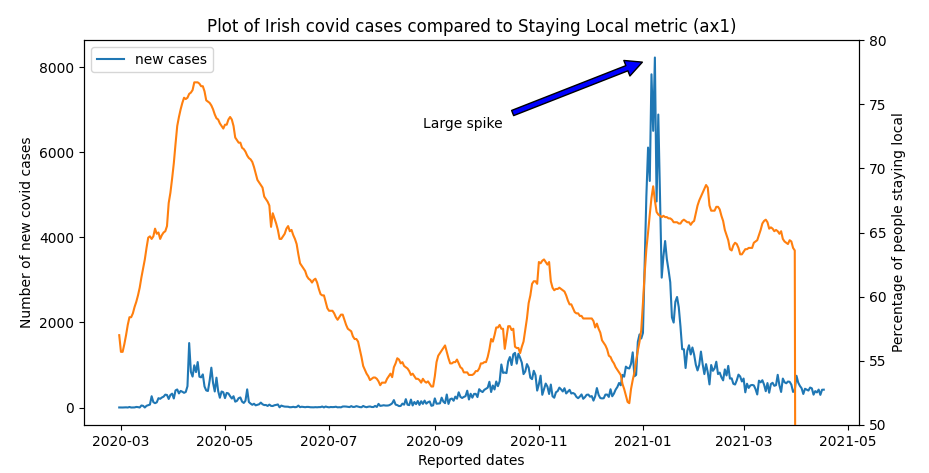
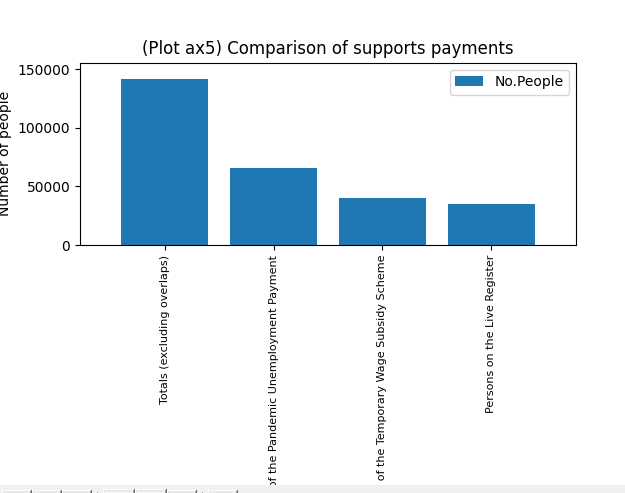
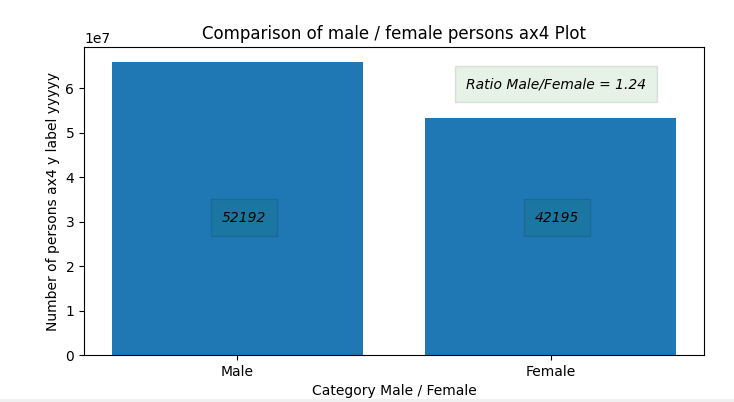


Figure 3 plot of Irish covid case and staying local % by date

****



# Insights

<<Point out at least 5 insights in bullet points>>

# References

<https://www.cso.ie/en/releasesandpublications/fp/fp-sli/covid-19insightmobilityduringthepandemicweekending31december2020/frequentlyaskedquestions/>

<https://data.cso.ie/table/SLI01>

The Staying Local Indicator (SLI) provides daily estimated percentages of county populations that have stayed local i.e. within 10km of their usual place of residence, averaged over the preceding seven days.

# Bibliography

|  |  |
| --- | --- |
| [1] | H. Ritchie, “Our World in Data,” [Online]. Available: #https://ws.cso.ie/public/api.jsonrpc?data=%7B%22jsonrpc%22:%222.0%22,%22method%22:%22PxStat.Data.Cube\_API.ReadDataset%22,%22params%22:%7B%22class%22:%22query%22,%22id%22:%5B%5D,%22dimension%22:%7B%7D,%22extension%22:%7B%22pivot%22:null,%22codes%22:false,. [Accessed 14 April 2021]. |
| [2] | https://data.cso.ie/, “https://data.cso.ie/,” Staying Local Indicator, [Online]. Available: https://data.cso.ie/. [Accessed 17 April 2021]. |
| [3] | C. I. -. I. Government, “COVID-19 Pandemic Unemployment Payment,” [Online]. Available: https://www.citizensinformation.ie/en/social\_welfare/social\_welfare\_payments/unemployed\_people/covid19\_pandemic\_unemployment\_payment.html. [Accessed 17 April 2021]. |
| [4] | I. Central Statistics Office, “Persons on the Live Register, in receipt of the Pandemic Unemployment Payment , in receipt of the Temporary Wage Subsidy Scheme and Total excluding overlaps,” Central Statistics Office, Ireland, [Online]. Available: https://data.cso.ie/. [Accessed 17 April 2021]. |