Merging JsonStat and GeoJson formatted data to create a GeoDataFrame, it’s visualisation, and writing it to an Esri Shapefile

An example from Ireland using Python

May 2019

Kevin McCormack
Dr. Mary Smyth
Sinead Phelan
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2. Data Repository - Statbank</td>
<td>3</td>
</tr>
<tr>
<td>3. Geographic data repository – Open Data Portal</td>
<td>3</td>
</tr>
<tr>
<td>4. JSON-stat format</td>
<td>3</td>
</tr>
<tr>
<td>5. GeoJSON</td>
<td>4</td>
</tr>
<tr>
<td>6. Python</td>
<td>4</td>
</tr>
<tr>
<td>6.1. Pandas DataFrame</td>
<td>4</td>
</tr>
<tr>
<td>6.2. GeoDataFrame</td>
<td>5</td>
</tr>
<tr>
<td>6.3. Python libraries</td>
<td>5</td>
</tr>
<tr>
<td>7. Python code</td>
<td>6</td>
</tr>
<tr>
<td>7.1. Installing the libraries</td>
<td>6</td>
</tr>
<tr>
<td>7.2. Importing and initialising the main python libraries</td>
<td>6</td>
</tr>
<tr>
<td>8. Setting current directory and downloading JsonStat file from the CSO’s Statbank API</td>
<td>7</td>
</tr>
<tr>
<td>9. Extracting the JsonStatCollection</td>
<td>8</td>
</tr>
<tr>
<td>10. Initialising the JsonStatCollection from the file</td>
<td>8</td>
</tr>
<tr>
<td>11. Selecting and extract dataset contained into collection</td>
<td>9</td>
</tr>
<tr>
<td>12. Exploring the dimensions of the JsonStatDataSet:</td>
<td>9</td>
</tr>
<tr>
<td>13. Accessing a value in the JsonStatDataSet, “dataset”</td>
<td>13</td>
</tr>
<tr>
<td>14. Transforming the JsonStatDataSet, “dataset”, into a pandas DataFrame</td>
<td>13</td>
</tr>
<tr>
<td>15. Renaming columns in a dataframe</td>
<td>15</td>
</tr>
<tr>
<td>16. Creating a GeoDataFrame directly from a GeoJson dataset on OSI’s open portal</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Title</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>17</td>
<td>Merging the Data and GeoData Frames</td>
</tr>
<tr>
<td>18</td>
<td>Creating additional GeoDataFrames from dfb</td>
</tr>
<tr>
<td>19</td>
<td>Writing a GeoDataFrame to an Esri shape file</td>
</tr>
<tr>
<td>20</td>
<td>Visualisation with python</td>
</tr>
<tr>
<td>21</td>
<td>Converting matplotlib plots from Python into interactive</td>
</tr>
<tr>
<td></td>
<td>Leaflet web maps.</td>
</tr>
</tbody>
</table>
1. Introduction

In this tutorial we will discuss how to extract and join statistical data from the CSO's online database, StatBank, with geographic data, namely polygons, extracted from the OSi open data portal to create a GeoDataFrame, it's visualisation, and write it to an Esri Shapefile.

2. Data Repository - Statbank

StatBank is the CSO's online database of Official Statistics. This database contains current and historical data series compiled from CSO statistical releases and is accessed at http://www.cso.ie/px/pxeirestat/statire/SelectTable/Omrade0.asp?Planguage=0.

3. Geographic data repository – Open Data Portal

The OSi provide Open Geographic Data is data that can be freely used, re-used and redistributed by anyone - subject only, at most, to the requirement that the source of the information is to attributed and accessed at https://data-osi.opendata.arcgis.com/.

4. JSON-stat format

The **JSON-stat format** is a simple lightweight JSON\(^1\) format for data dissemination. It is based in a cube model that arises from the evidence that the most common form of data dissemination is the tabular form. In this cube model, datasets are organized in dimensions. Dimensions are organized in categories.

Data dissemination is not the business of a few anymore. Even though the **JSON-stat format** can be the perfect companion for the **open data initiatives** of National Statistical Offices, it is suitable for all kinds of data disseminators because it has been designed with simplicity in mind.

---

\(^1\) **JSON** (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the [JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999](https://www.ecma-international.org/publications/standards/Ecma-262.htm). JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.
5. **GeoJSON**

GeoJSON is a format for encoding a variety of geographic data structures. GeoJSON supports the following geometry types:

- Point,
- LineString,
- Polygon,
- MultiPoint,
- MultiLineString, and
- MultiPolygon.

Geometric objects with additional properties are Feature objects. Sets of features are contained by FeatureCollection objects.

6. **Python**

Python is an interpreted, high-level, general-purpose programming language.

Python can be downloaded and installed from either

https://www.python.org/downloads/, or

https://www.anaconda.com/

6.1 **Pandas DataFrame**

A Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the **data, rows**, and **columns**.
6.2 GeoDataFrame

A GeoDataFrame object is a pandas.DataFrame that has a column with geometry.

6.3 Python libraries

Within Python there are numerous libraries which are collections of functions and methods that allows one to perform many actions without writing your own code.

In this tutorial several libraries are installed, imported and used.
7. Python code

There are several steps in the process.

7.1 Installing the libraries

In this tutorial the following libraries are installed.

```bash
!pip install jsonstat.py
!pip install pyjstat
```

It should be noted that to when installing the geopandas library one has install several dependencies libraries. A number of these libraries they may not install directly and one may have to go to this website and download the .whl file and manually load using Anaconda prompt.  [https://www.lfd.uci.edu/~gohlke/pythonlibs/#shapely](https://www.lfd.uci.edu/~gohlke/pythonlibs/#shapely)

```bash
!pip install numpy
!pip install GDAL  # had to be manually loaded
!pip install pyshp
!pip install Shapley  # had to be manually loaded
!pip install Fiona  # had to be manually loaded
!pip install geopy
!pip install pyproj
!pip install geopandas  # had to be manually loaded
!pip install descartes
```

If one wishes to visualise a GeoDataFrame then install these libraries.

```bash
!pip install matplotlib
!pip install mplleaflet
```

7.2 Importing and initialising the main python libraries

To activate the libraries, one must import and initialise them.
import geopandas as gpd
import pandas as pd  # using panda to convert jsonstat data set to pandas dataframe
import os  # The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, etc.

from pyjstat import pyjstat
import numpy as np
import shapefile as shp
import jsonstat  # import jsonstat.py package
import matplotlib.pyplot as plt  # for plotting
import requests
import seaborn as sns

8. Setting current directory and downloading JsonStat file from the CSO’s Statbank API

The following code allows one to identify and set and current working directory

os.getcwd()  # current working directory 'C:\XXXX'
os.chdir('C:\XXXX\Python')  # changing current directory

We will now download the JsonStat QLF07 file, also known as a collection, from CSO Statbank’s API, which contains data for Persons aged 15 years and over in Employment by Sex, NACE Rev 2 Economic Sector, Region and Quarter obtained from the CSO’s Quarterly Labour Force Survey

First, we will define a url and file_name as follows

url = 'https://www.cso.ie/StatbankServices/StatbankServices.svc/jsonservice/responseinstance/QLF07'
file_name = "QLF07.json"
9. **Extracting the JsonStatCollection**

We will extract the JsonStatCollection and create a directory for data

```python
jsonstat.from_url(url, file_name)
file_path = (os.path.join('C:\\XXXX', file_name))
```

10. **Initialising the JsonStatCollection from the file**

We will now Initialise JsonStatCollection from the file and print the list of dataset contained into the collection.

```python
collection = jsonstat.from_file(file_path)
```

To identify the number of dataset contained in this collection

```python
len(collection)
```

The value ‘1’ is returned indicating that there is one dataset contained in this collection and this can be confirmed by running the command,

```python
collection
```

which provides this output,

```text
JsonstatCollection contains the following JsonStatDataSet:

<table>
<thead>
<tr>
<th>pos</th>
<th>dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'dataset'</td>
</tr>
</tbody>
</table>
```
11. Selecting and extract dataset contained into collection

We select, and extract dataset contained in the collection and name it QLF07 as follows:

\[
QLF07 = collection.dataset(0)
\]

To understand what is contained in the dataset simply run the code,

\[
QLF07
\]

Which provides the following output.

name: 'dataset'
lable: 'Persons aged 15 years and over in Employment by Sex, NACE Rev 2 Economic Sector, Region and Quarter'
source: 'Persons aged 15 years and over in Employment by Sex, NACE Rev 2 Economic Sector, Region and Quarter'
size: 13122

+---------------------------------+-----------------+--------+-------+------+
| pos | id             | label                                             | size | role |
+---------------------------------+-----------------+-----------------+--------+-------+------+
| 0   | Sex            | Sex                                                     | 3     |      |     |
| 1   | NACE Rev 2 Economic Sector | NACE Rev 2 Economic Sector | 18    |      |     |
| 2   | Region         | Region                                                  | 9     |      |     |
| 3   | Quarter        | Quarter                                                 | 27    | time |     |
| 4   | Statistic      | Statistic                                              | 1     | metric |     |
+---------------------------------+-----------------+-----------------+--------+-------+------+

12. Exploring the dimensions of the JsonStatDataSet:

One can obtain some details about the dimensions, i.e categories within a dimension as follows.

\[
QLF07.dimension('dataset')
\]

Which provides the following output with the name of the dimensions identified.
JsonStatException: "dataset 'dataset': unknown dimension 'dataset' know dimensions
ids are:
Sex, NACE Rev 2 Economic Sector, Region, Quarter, Statistic"

To investigate the dimension region, for example, use the following command.

QLF07.dimension('Region')

Which returns the following.

+-------------------------------+
<table>
<thead>
<tr>
<th>pos</th>
<th>idx</th>
<th>label</th>
</tr>
</thead>
</table>
+-------------------------------+
| 0   | '-'    | 'State'      |
| 1   | 'IE041'| 'Border'     |
| 2   | 'IE042'| 'West'       |
| 3   | 'IE051'| 'Mid-West'   |
| 4   | 'IE052'| 'South-East' |
| 5   | 'IE053'| 'South-West' |
| 6   | 'IE061'| 'Dublin'     |
| 7   | 'IE062'| 'Mid-East'   |
| 8   | 'IE063'| 'Midland'    |
+-------------------------------+

The following code provides details on the Nace Rev 2 Economic Sector dimension.

QLF07.dimension('NACE Rev 2 Economic Sector')

It can also be referenced as dimension 1, as follow.

QLF07.dimension(1)

Which returns the following.

# dimension 1
We continue checking the other dimensions, with

QLF07.dimension("Sex")

returning:

<table>
<thead>
<tr>
<th>pos</th>
<th>idx</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>'Both sexes'</td>
</tr>
<tr>
<td>1</td>
<td>'1'</td>
<td>'Male'</td>
</tr>
<tr>
<td>2</td>
<td>'2'</td>
<td>'Female'</td>
</tr>
</tbody>
</table>

and
And for the Statistic dimension, number 4,

QLF07.dimension(4)

returns
13. **Accessing a value in the JsonStatDataSet, “dataset”**

To access a value in the dataset one references the dimension values such as:

```
QLF07.data(Region = 'IE041', Sex = '1', Quarter = '2018Q3')
```

Which returns

```
JsonStatValue(idx=4427, value=97.5, status=None)
```

It should be noted that the dimension name NACE Rev 2 Economic Sector cannot be used as it contains spaces, simply use the dimension number instead.

14. **Transforming the JsonStatDataSet, “dataset”, into a pandas DataFrame**

The following code transforms the JsonStatDataSet, “dataset”, into a pandas DataFrame

```
df_QLF07 = QLF07.to_data_frame(content='idx')
```

Note that the `content = 'idx'` is used to grab the geography category values, which we will call GEOGID, from the regional dimension

It is also possible to grab the label, by replacing `content='idx'` with `content = 'label'`

```
df_QLF07A = QLF07.to_data_frame(content='label')
```

Again, one can display the dataframe.

```
display(df_QLF07)
```

Which provides the following output.
<table>
<thead>
<tr>
<th>Sex</th>
<th>NACE Rev 2 Economic Sector</th>
<th>Region</th>
<th>Quarter</th>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>2012Q1</td>
<td>QLF07C01</td>
<td>1863.2</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2012Q2</td>
<td>QLF07C01</td>
<td>1878.0</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>2012Q3</td>
<td>QLF07C01</td>
<td>1887.0</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>2012Q4</td>
<td>QLF07C01</td>
<td>1893.6</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>2013Q1</td>
<td>QLF07C01</td>
<td>1892.0</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>2013Q2</td>
<td>QLF07C01</td>
<td>1926.3</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>2013Q3</td>
<td>QLF07C01</td>
<td>1961.8</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>2013Q4</td>
<td>QLF07C01</td>
<td>1971.0</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
<td>2014Q1</td>
<td>QLF07C01</td>
<td>1950.7</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>-</td>
<td>2014Q2</td>
<td>QLF07C01</td>
<td>1970.3</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
<td>2014Q3</td>
<td>QLF07C01</td>
<td>2008.9</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>-</td>
<td>2014Q4</td>
<td>QLF07C01</td>
<td>2025.2</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>-</td>
<td>2015Q1</td>
<td>QLF07C01</td>
<td>2014.4</td>
</tr>
<tr>
<td>13</td>
<td>-</td>
<td>-</td>
<td>2015Q2</td>
<td>QLF07C01</td>
<td>2049.7</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>-</td>
<td>2015Q3</td>
<td>QLF07C01</td>
<td>2079.9</td>
</tr>
<tr>
<td>15</td>
<td>-</td>
<td>-</td>
<td>2015Q4</td>
<td>QLF07C01</td>
<td>2085.4</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>-</td>
<td>2016Q1</td>
<td>QLF07C01</td>
<td>2080.8</td>
</tr>
<tr>
<td>17</td>
<td>-</td>
<td>-</td>
<td>2016Q2</td>
<td>QLF07C01</td>
<td>2126.7</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>-</td>
<td>2016Q3</td>
<td>QLF07C01</td>
<td>2158.0</td>
</tr>
<tr>
<td>19</td>
<td>-</td>
<td>-</td>
<td>2016Q4</td>
<td>QLF07C01</td>
<td>2163.5</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>-</td>
<td>2017Q1</td>
<td>QLF07C01</td>
<td>2158.4</td>
</tr>
<tr>
<td>21</td>
<td>-</td>
<td>-</td>
<td>2017Q2</td>
<td>QLF07C01</td>
<td>2180.9</td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>-</td>
<td>2017Q3</td>
<td>QLF07C01</td>
<td>2206.5</td>
</tr>
<tr>
<td>23</td>
<td>-</td>
<td>-</td>
<td>2017Q4</td>
<td>QLF07C01</td>
<td>2230.8</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>-</td>
<td>2018Q1</td>
<td>QLF07C01</td>
<td>2220.7</td>
</tr>
<tr>
<td>25</td>
<td>-</td>
<td>-</td>
<td>2018Q2</td>
<td>QLF07C01</td>
<td>2255.0</td>
</tr>
<tr>
<td>26</td>
<td>-</td>
<td>-</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>2273.2</td>
</tr>
<tr>
<td>27</td>
<td>-</td>
<td>-</td>
<td>2012Q1</td>
<td>QLF07C01</td>
<td>145.1</td>
</tr>
<tr>
<td>28</td>
<td>-</td>
<td>-</td>
<td>2012Q2</td>
<td>QLF07C01</td>
<td>146.3</td>
</tr>
<tr>
<td>29</td>
<td>-</td>
<td>-</td>
<td>2012Q3</td>
<td>QLF07C01</td>
<td>145.3</td>
</tr>
</tbody>
</table>

One can display the dataframe with the labels.

```python
display(df_QLF07A)
```
15. **Renaming columns in a dataframe**

To confirm the column list in the dataframes use the following code.

```python
df_QLF07.columns
```

which returns,

```
Out[79]: Index(['Sex', 'NACE Rev 2 Economic Sector', 'Region', 'Quarter', 'Statistic', 'Value'], dtype='object')
```

and,

```python
df_QLF07
```

This code can be used to confirm the column list in the dataframes.
df_QLF07A.columns

which returns

Out[80]: Index(['Sex', 'NACE Rev 2 Economic Sector', 'Region', 'Quarter', 'Statistic', 'Value'], dtype='object')

We can now rename columns Region to NUTS3 in df_QLF07 dataframe and to NUTS3NAME in df_QLF07A dataframe, in order to join with the GeoJson file below

df_QLF07.rename(columns={'Region':'NUTS3'}, inplace=True)
df_QLF07A.rename(columns={'Region':'NUTS3NAME'}, inplace=True)

We will now list the of columns in each datatframe to confirm the change of name.

df_QLF07.columns

returns,

Out[83]: Index(['Sex', 'NACE Rev 2 Economic Sector', 'NUTS3', 'Quarter', 'Statistic', 'Value'], dtype='object')

and,

df_QLF07A.columns

returns,

Out[84]: Index(['Sex', 'NACE Rev 2 Economic Sector', 'NUTS3NAME', 'Quarter', 'Statistic', 'Value'], dtype='object')
16.  Creating a GeoDataFrame directly from GeoJson dataset on OSi’s open portal

We will now create a dataframe directly from a GeoJson data for the NUTS3 Regions from the Ordnance Survey Ireland’s (OSi) open data portal

Again we will define a url as follows:

```
url1 = "http://data-os.opendata.arcgis.com/datasets/8e1da9ca81cb478d8146580b130abe08_2.geojson"
```

Then create the data frame.

```
dfa = gpd.read_file(url1)
```

To ensure that the a GeoDataFrame has been created, we can simply run the code.

```
type(dfa)
```

which returns,

```
geopandas.geodataframe.GeoDataFrame
```

We can list the columns in the GeoDataFrame,

```
dfa.columns
```

which returns,

```
Index(['OBJECTID', 'NUTS1', 'NUTS1NAME', 'NUTS2', 'NUTS2NAME', 'NUTS3', 'NUTS3NAME', 'GUID', 'Shape__Area', 'Shape__Length', 'geometry'], dtype='object')
```

We can explore this GeoDataFrame by referencing the columns.

```
dfa.GUID
```
returns,

<table>
<thead>
<tr>
<th></th>
<th>GUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>B1A65D7C-1984-4A87-AD58-0E846812C992</td>
</tr>
<tr>
<td>1</td>
<td>A69CA800-8D87-4920-A7C1-50426A1D39B4</td>
</tr>
<tr>
<td>2</td>
<td>42C5C2A5-2D71-4BD1-BDB5-BD7D3198CD78</td>
</tr>
<tr>
<td>3</td>
<td>27C93D4E-AD0E-4B0C-8FA8-3566AEEDA5CC</td>
</tr>
<tr>
<td>4</td>
<td>604546A1-A856-4B9B-AD46-E88B27C27155</td>
</tr>
<tr>
<td>5</td>
<td>F97E459B-57ED-49C0-8A2B-2BC1C7F08E88</td>
</tr>
<tr>
<td>6</td>
<td>8E4862CC-7E43-4BF5-A4EF-B2D5ECBA61EF</td>
</tr>
<tr>
<td>7</td>
<td>B26C8BAA-F3C5-49A9-B74E-D7FED1823E65</td>
</tr>
</tbody>
</table>

Name: GUID, dtype: object

With

```
dfa.NUTS3
```

returns,

<table>
<thead>
<tr>
<th></th>
<th>NUTS3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IE041</td>
</tr>
<tr>
<td>1</td>
<td>IE042</td>
</tr>
<tr>
<td>2</td>
<td>IE051</td>
</tr>
<tr>
<td>3</td>
<td>IE052</td>
</tr>
<tr>
<td>4</td>
<td>IE053</td>
</tr>
<tr>
<td>5</td>
<td>IE061</td>
</tr>
<tr>
<td>6</td>
<td>IE062</td>
</tr>
<tr>
<td>7</td>
<td>IE063</td>
</tr>
</tbody>
</table>

Name: NUTS3, dtype: object

While,

```
dfa.NUTS3NAME
```

returns,

<table>
<thead>
<tr>
<th></th>
<th>NUTS3NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Border</td>
</tr>
<tr>
<td>1</td>
<td>West</td>
</tr>
</tbody>
</table>
We will create a new GeoDataFrame by merging the Data and Geodata frames on the NUTS3 column.

```python
dfb = dfa.merge(df_QLF07, on='NUTS3')
```

We will now list the of columns in the new GeoDataFrame to confirm the merge was successful.

```python
dfb.columns
```

which returns,

```python
Index(['OBJECTID', 'NUTS1', 'NUTS1NAME', 'NUTS2', 'NUTS2NAME', 'NUTS3', 'NUTS3NAME', 'GUID', 'Shape__Area', 'Shape__Length', 'geometry', 'Sex', 'NACE Rev 2 Economic Sector', 'Quarter', 'Statistic', 'Value'], dtype='object')
```

We can use head and tail commands to quickly view the new GeoDataFrame, `dfb`. 

```python
dfb.head()
dfb.tail()
```
The head command,

```python
dfb.head()
```

will output the 1st five lines of the GeoDataFrame as follows, (not all columns are shown by default):

<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>NUTS1</th>
<th>NUTS1NAME</th>
<th>NUTS2</th>
<th>NUTS2NAME</th>
<th>NUTS3</th>
<th>NUTS3NAME</th>
<th>GUID</th>
<th>Shape_Area</th>
<th>Shape_Length</th>
<th>geometry</th>
<th>Quarter</th>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>IE</td>
<td>IE04</td>
<td>Northern and Western IE04</td>
<td>Border</td>
<td>B1A65D7C-1984-4A87-AD58-0E846812C992</td>
<td>1.594889</td>
<td>31.557961</td>
<td>(POLYGON (((-7.25362176708799 55.4458025525653,...</td>
<td>-</td>
<td>2012Q1</td>
<td>QLF07C01</td>
<td>145.1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>IE</td>
<td>IE04</td>
<td>Northern and Western IE04</td>
<td>Border</td>
<td>B1A65D7C-1984-4A87-AD58-0E846812C992</td>
<td>1.594889</td>
<td>31.557961</td>
<td>(POLYGON (((-7.25362176708799 55.4458025525653,...</td>
<td>-</td>
<td>2012Q2</td>
<td>QLF07C01</td>
<td>146.3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>IE</td>
<td>IE04</td>
<td>Northern and Western IE04</td>
<td>Border</td>
<td>B1A65D7C-1984-4A87-AD58-0E846812C992</td>
<td>1.594889</td>
<td>31.557961</td>
<td>(POLYGON (((-7.25362176708799 55.4458025525653,...</td>
<td>-</td>
<td>2012Q3</td>
<td>QLF07C01</td>
<td>145.3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>IE</td>
<td>IE04</td>
<td>Northern and Western IE04</td>
<td>Border</td>
<td>B1A65D7C-1984-4A87-AD58-0E846812C992</td>
<td>1.594889</td>
<td>31.557961</td>
<td>(POLYGON (((-7.25362176708799 55.4458025525653,...</td>
<td>-</td>
<td>2012Q4</td>
<td>QLF07C01</td>
<td>147.7</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>IE</td>
<td>IE04</td>
<td>Northern and Western IE04</td>
<td>Border</td>
<td>B1A65D7C-1984-4A87-AD58-0E846812C992</td>
<td>1.594889</td>
<td>31.557961</td>
<td>(POLYGON (((-7.25362176708799 55.4458025525653,...</td>
<td>-</td>
<td>2013Q1</td>
<td>QLF07C01</td>
<td>150.9</td>
</tr>
</tbody>
</table>

and, the tail command,

```python
dfb.tail()
```

will output the last five lines of the GeoDataFrame as follows, (not all columns are shown by default):

<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>NUTS1</th>
<th>NUTS1NAME</th>
<th>NUTS2</th>
<th>NUTS2NAME</th>
<th>NUTS3</th>
<th>NUTS3NAME</th>
<th>GUID</th>
<th>Shape_Area</th>
<th>Shape_Length</th>
<th>geometry</th>
<th>NACE Rev 2 Economic Sector</th>
<th>Quarter</th>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11659</td>
<td>8</td>
<td>IE</td>
<td>IE06</td>
<td>Eastern and Midland IE06</td>
<td>Midlands</td>
<td>B26C8BAA-F3C5-49A9-B74E-D7FED1823E5</td>
<td>0.896836</td>
<td>7.564148</td>
<td>POLYGON (((-7.64673135464132 53.9415530759351,...</td>
<td>2</td>
<td>ZXD400</td>
<td>2017Q3</td>
<td>QLF07C01</td>
<td>NaN</td>
</tr>
<tr>
<td>11660</td>
<td>8</td>
<td>IE</td>
<td>IE06</td>
<td>Eastern and Midland IE06</td>
<td>Midlands</td>
<td>B26C8BAA-F3C5-49A9-B74E-D7FED1823E5</td>
<td>0.896836</td>
<td>7.564148</td>
<td>POLYGON (((-7.64673135464132 53.9415530759351,...</td>
<td>2</td>
<td>ZXD400</td>
<td>2017Q4</td>
<td>QLF07C01</td>
<td>NaN</td>
</tr>
<tr>
<td>11661</td>
<td>8</td>
<td>IE</td>
<td>IE06</td>
<td>Eastern and Midland IE06</td>
<td>Midlands</td>
<td>B26C8BAA-F3C5-49A9-B74E-D7FED1823E5</td>
<td>0.896836</td>
<td>7.564148</td>
<td>POLYGON (((-7.64673135464132 53.9415530759351,...</td>
<td>2</td>
<td>ZXD400</td>
<td>2018Q1</td>
<td>QLF07C01</td>
<td>NaN</td>
</tr>
<tr>
<td>11662</td>
<td>8</td>
<td>IE</td>
<td>IE06</td>
<td>Eastern and Midland IE06</td>
<td>Midlands</td>
<td>B26C8BAA-F3C5-49A9-B74E-D7FED1823E5</td>
<td>0.896836</td>
<td>7.564148</td>
<td>POLYGON (((-7.64673135464132 53.9415530759351,...</td>
<td>2</td>
<td>ZXD400</td>
<td>2018Q2</td>
<td>QLF07C01</td>
<td>NaN</td>
</tr>
<tr>
<td>11663</td>
<td>8</td>
<td>IE</td>
<td>IE06</td>
<td>Eastern and Midland IE06</td>
<td>Midlands</td>
<td>B26C8BAA-F3C5-49A9-B74E-D7FED1823E5</td>
<td>0.896836</td>
<td>7.564148</td>
<td>POLYGON (((-7.64673135464132 53.9415530759351,...</td>
<td>2</td>
<td>ZXD400</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>NaN</td>
</tr>
</tbody>
</table>
18. Creating additional GeoDataFrames from dfb

We will now create individual GeoDataFrames for males, females and total, by parsing the GeoDataFrame, dfb, where:

- **Sex:** 1 = Males, 2 = Females, - = Total,
- **NACE Rev 2 Economic Sector** = F, Construction, and
- **Quarter** = 2018Q3

\[
\begin{align*}
\text{dfbm} & = \text{dfb.loc[(dfb['Sex'] == '1') & (dfb['NACE Rev 2 Economic Sector'] == 'F') & (dfb['Quarter'] == '2018Q3')]} \\
\text{dfbf} & = \text{dfb.loc[(dfb['Sex'] == '2') & (dfb['NACE Rev 2 Economic Sector'] == 'F') & (dfb['Quarter'] == '2018Q3')]} \\
\text{dfbt} & = \text{dfb.loc[(dfb['Sex'] == '-') & (dfb['NACE Rev 2 Economic Sector'] == 'F') & (dfb['Quarter'] == '2018Q3')]} \\
\end{align*}
\]

We can output the 1st five lines of each GeoDataFrame, for male, female and total.

\[
\text{dfbm.head()} \\
\begin{align*}
\text{OBJECTID} & \quad \text{NUTS1} & \quad \text{NUTS1 NAME} & \quad \text{NUTS2} & \quad \text{NUTS2 NAME} & \quad \text{NUTS3} & \quad \text{NUTS3 NAME} & \quad \text{GUID} & \quad \text{Shape__Area} & \quad \text{Shape__Length} & \quad \text{geometry} & \quad \text{Sex} & \quad \text{Quarter} & \quad \text{Statistic} & \quad \text{Value} \\
566 & 1 & IE Ireland & IE04 & Northern and Western & IE041 & Border & B1A65D7C-1984-4A87-AD58-0E846812C992 & 1.594889 & 31.557961 & (POLYGON (((7.25362176708799 55.4458025525653, )),... & \text{1} & 2018Q3 & QLF07C01 & 13.9 \\
2024 & 2 & IE Ireland & IE04 & Northern and Western & IE042 & West & A69CA800-BD87-4920-7C11-50426A1D39B4 & 1.940981 & 38.200397 & (POLYGON (((9.524108481921839 53.068274381794, )),... & \text{1} & 2018Q3 & QLF07C01 & 13.3 \\
3482 & 3 & IE Ireland & IE05 & Southern & IE051 & Mid-West & 42C5C2A5-2D71-4B01-8DB5-BD7D3198CD78 & 1.396164 & 11.020339 & (POLYGON (((9.935623142260511 52.5611963634314, )),... & \text{1} & 2018Q3 & QLF07C01 & 14.5 \\
4940 & 4 & IE Ireland & IE05 & Southern & IE052 & South-East & 27C93D4E-AD9E-4BC0-7FA8-3566AED5CC & 0.951352 & 11.412129 & (POLYGON (((7.76097866504068 51.9451840492602, )),... & \text{1} & 2018Q3 & QLF07C01 & 12.3 \\
6398 & 5 & IE Ireland & IE05 & Southern & IE053 & South-West & 604548A1-A856-4B9B-AD4E-E88827C27155 & 1.610451 & 32.378775 & (POLYGON (((9.472000632876481 51.4533061425, )),... & \text{1} & 2018Q3 & QLF07C01 & 20.5 \\
\end{align*}
\]
dfbf.head()

<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>NUTS1</th>
<th>NUTS1NAME</th>
<th>NUTS2</th>
<th>NUTS2NAME</th>
<th>NUTS3</th>
<th>NUTS3NAME</th>
<th>GUID</th>
<th>Shape_Area</th>
<th>Shape_Length</th>
<th>geometry</th>
<th>Sex</th>
<th>Quarter</th>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1052</td>
<td>1</td>
<td>IE</td>
<td>Ireland</td>
<td>Northern and Western</td>
<td>IE04</td>
<td>Border</td>
<td>B1A65D7C-1984-A87-AD58-0E846812C992</td>
<td>1.594889</td>
<td>31.557961</td>
<td>(POLYGON (((7.25362176708799 55.4458025525653,...</td>
<td>2</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>NaN</td>
</tr>
<tr>
<td>2510</td>
<td>2</td>
<td>IE</td>
<td>Ireland</td>
<td>Northern and Western</td>
<td>IE04</td>
<td>West</td>
<td>A69CA800-8D87-4920-ATC1-50426A1D39B4</td>
<td>1.940981</td>
<td>38.200397</td>
<td>(POLYGON (((9.524108481921839 53.66827438179,...</td>
<td>2</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>NaN</td>
</tr>
<tr>
<td>3968</td>
<td>3</td>
<td>IE</td>
<td>Ireland</td>
<td>Southern</td>
<td>IE05</td>
<td>Mid West</td>
<td>42C522A5-2D71-4BD1-BDB5-B7D7D198CD759 1.396164</td>
<td>11.020333</td>
<td>(POLYGON (((9.935623142260511 52.561196364314...</td>
<td>2</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>5426</td>
<td>4</td>
<td>IE</td>
<td>Ireland</td>
<td>Southern</td>
<td>IE05</td>
<td>South-East</td>
<td>27C93D4E-AD5E-4B0C-8FA8-3566AED5CC 0.951352</td>
<td>11.412129</td>
<td>(POLYGON (((7.76097866504068 51.9451840492602,...</td>
<td>2</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>6884</td>
<td>5</td>
<td>IE</td>
<td>Ireland</td>
<td>Southern</td>
<td>IE05</td>
<td>South-West</td>
<td>604546A1-A856-4B9B-AD46-E88B27C27155 1.610451</td>
<td>32.378775</td>
<td>(POLYGON (((9.4720063287648151.4533061425, ...</td>
<td>2</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>NaN</td>
<td></td>
</tr>
</tbody>
</table>

dfbt.head()

<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>NUTS1</th>
<th>NUTS1NAME</th>
<th>NUTS2</th>
<th>NUTS2NAME</th>
<th>NUTS3</th>
<th>NUTS3NAME</th>
<th>GUID</th>
<th>Shape_Area</th>
<th>Shape_Length</th>
<th>geometry</th>
<th>Sex</th>
<th>Quarter</th>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>1</td>
<td>IE</td>
<td>Ireland</td>
<td>Northern and Western</td>
<td>IE04</td>
<td>Border</td>
<td>B1A65D7C-1984-A87-AD58-0E846812C992</td>
<td>1.594889</td>
<td>31.557961</td>
<td>(POLYGON (((7.25362176708799 55.4458025525653,...</td>
<td>-</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>14.9</td>
</tr>
<tr>
<td>1538</td>
<td>2</td>
<td>IE</td>
<td>Ireland</td>
<td>Northern and Western</td>
<td>IE04</td>
<td>West</td>
<td>A69CA800-8D87-4920-ATC1-50426A1D39B4 1.940981</td>
<td>38.200397</td>
<td>(POLYGON (((9.524108481921839 53.66827438179,...</td>
<td>-</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td>2996</td>
<td>3</td>
<td>IE</td>
<td>Ireland</td>
<td>Southern</td>
<td>IE05</td>
<td>Mid-West</td>
<td>42C522A5-2D71-4BD1-BDB5-B7D7D198CD759 1.396164</td>
<td>11.020333</td>
<td>(POLYGON (((9.935623142260511 52.561196364314...</td>
<td>-</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>15.2</td>
<td></td>
</tr>
<tr>
<td>4454</td>
<td>4</td>
<td>IE</td>
<td>Ireland</td>
<td>Southern</td>
<td>IE05</td>
<td>South-West</td>
<td>27C93D4E-AD5E-4B0C-8FA8-3566AED5CC 0.951352</td>
<td>11.412129</td>
<td>(POLYGON (((7.76097866504068 51.9451840492602,...</td>
<td>-</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>12.7</td>
<td></td>
</tr>
<tr>
<td>5912</td>
<td>5</td>
<td>IE</td>
<td>Ireland</td>
<td>Southern</td>
<td>IE05</td>
<td>South-West</td>
<td>604546A1-A856-4B9B-AD46-E88B27C27155 1.610451</td>
<td>32.378775</td>
<td>(POLYGON (((9.4720063287648151.4533061425, ...</td>
<td>-</td>
<td>2018Q3</td>
<td>QLF07C01</td>
<td>21.8</td>
<td></td>
</tr>
</tbody>
</table>
19. Writing a GeoDataFrame to an Esri shape file

We will now write GeoDataFrame for containing the data for males, dfbm, to an Esri shape file for visualisation, as follows:

```
dfbm.to_file('NUTS3__Generalised_20mA', driver='ESRI Shapefile')
```

A NUTS3__Generalised_20mA folder is created in the working directory 'C:\XXXX\Python'.

20. Visualisation with python

One can also visualise a GeoDataFrame in Python using matplotlib.

Import the library.

```
import matplotlib.pyplot as plt
```

One can then preview the GeoDataFrame, dfa, as follows:

```
dfa.plot()
```

Which returns the NUTS3 outlines, as it contains no data.

We can now visualise the GeoDataFrame, dfbm, which contains data on the number of Males in the Construction Industry for Quarter 3 of 2018..
We set the column that will be called to be visualised on the map.

```python
variable = 'Value'
```

Next, we set up the figure and axis.

```python
fig, ax = plt.subplots(1, figsize=(10,6))
```

We now add layer of polygons on the axis

```python
dfbmpp = dfbm.plot(column=variable,
                   ax=ax,
                   alpha = 0.8,  #intensity of colour of a polygon
                   cmap='Reds', #colour of the polygon
                   legend='True',
                   linewidth=0.3,
                   edgecolor='0.8',
                   label = 'Males')
```

We will also turn off the axis seen in Figure 1, as follows.

```python
ax.axis('off')
```

A figure title is added as follows,

```python
ax.set_title('Numbers of Males at work in the Construction Industry, Q3 2018')
```

and a legend title is set.

```python
ax.legend(title=('000'))
```
Now we can view the map.

```
ax.figure
```

which outputs:

![Map of Ireland with a color gradient indicating numbers of males at work in the Construction Industry, Q3 2018](image)

**Fig 2:** *Numbers of Males at work in the Construction Industry, Q3 2018*

21 Converting matplotlib plots from Python into interactive Leaflet web maps.

One can convert matplotlib plots from Python into interactive Leaflet web maps using the `mplleaflet` library.
Leaflet does not readily support the use of legends and requires very dark Choropleth map colours. We will return the code above leaving out the legend and changing the colour to ‘Dark2_r’ to produce a new `ax.figure`.

```python
variable = 'Value'
fig, ax = plt.subplots(1, figsize=(10,6))
dfbmpp = dfbm.plot(column=variable,
                   ax=ax,
                   alpha = 0.8,  #intensity of colour of a polygon
cmap='Dark2_r',  #colour of the polygon
legend='True',
linewidth=0.3,
edgecolor='0.8',
label = 'Males')
ax.axis('off')
ax.set_title('Numbers of Males at work in the Construction Industry, Q3 2018')
ax.figure
```

![Map of Ireland with color legend showing numbers of males at work in the construction industry, Q3 2018](image)

**Fig 2:** 'Numbers of Males at work in the Construction Industry, Q3 2018'

We now import the mplleaflet library,
import mplleaflet

and Output _map.html file and display it in your browser.

mplleaflet.show(fig=ax.figure)

**Fig 3**: html map, displayed in browser.