**Sarah Howlett  
Course: Certificate in Introductory Data Analytics   
Title: Data Analysis and Visualization on real-world Dataset**

GitHub repository URL: <https://github.com/howletts/UCDPA_SarahHowlett>

The repo contains 3 python scripts: import\_data.py, analyze\_data.py, and visualize\_data.py

A data folder containing the sqlite db, create table .sql file and pickles of the dataset.  
A visualizations folder containing the png files produced by running the python scripts.

**Real World Scenario**

The 3 datasets from Failte Ireland, Activities, Accommodation and Attractions were chosen by searching Irelands open data portal https://data.gov.ie/.

1. The ‘Activities’ dataset which consists of a collection of Activities. The **API was queried** with a filter on the search for records who’s tag included surfing, kitesurfing and windsurfing as this was the data of interest and due to a limit of 50 records being returned without subscribing.  
<https://failteireland.azure-api.net/opendata-api/v1/activities>  
  
2. The ‘Accommodation’ dataset which consists of a collection of Accommodations that have been quality approved by Fáilte Ireland and includes B&Bs, Caravan and Camping, Guesthouses, Hostels, Hotels and Self-catering. In this case the **csv was import to a DataFrame** in order to retrieve the full dataset.  
<https://failteireland.azure-api.net/opendata-api/v1/accommodation/csv>

3. The ‘Attractions’ dataset which consists of a collection of Tourist Attractions. In this case the **csv was downloaded** in order to retrieve the full dataset. <https://failteireland.azure-api.net/opendata-api/v1/attractions/csv>

Other reference data:  
  
4. Using SQLite Developer a demo database ‘sqlliteDB\_ucdproj.db’ was created with just one simple reference table COUNTY\_PROVINCE\_LINK to demonstrate **importing data from a Relational Database**. The database and create table SQL can be found in the data folder in the project repository/zip.

5. **Web data was scrapped** from a Wikipedia table to get areas designated as cities.  
<https://en.wikipedia.org/wiki/List_of_cities,_boroughs_and_towns_in_the_Republic_of_Ireland>

**Importing Data**

The **'make\_api\_call’** **reusable function** was defined to call an API given a URL as a parameter and load the returned json data into a DataFrame. It uses the Requests package to make the call and was used here to retrieve the Activities dataset filtered by records that had surfing, windsurfing and kitesurfing in the tag fields. The URL listed above was appended with this string to narrow the search.

**?$filter=search.ismatch('Surfing','tags') "** \  
**"and search.ismatch('Kitesurfing','tags') and search.ismatch('Windsurfing','tags')**

Similarly ‘import\_csv’ was defined to **import a csv file into a DataFrame**, given its URL. This function was used to import the full Accommodation and Attractions dataset.

The create\_engine function fromsqlalchemy package was used to create a database engine in order to **query the** **relational database** ‘sqlliteDB\_ucdproj.db’**.** Using the pandas read\_sql\_query function, passing it a select statement with a where clause, only records with a country of ROI where included (those from UK were excluded).

The pandas read\_html function was used to **scrape the data** from a wikipedia webpage. There were several tables on the webpage, the third dictionary in the list return was the one of interest here and was extracted using [2] on the list.

Note: Once the data was imported into DataFrames **python** **pickles** were generated for these.   
They can be found in the data folder in the project repository/zip. These were useful during analysis to save time and avoid calling endpoints or scrapping html each time the code was run.

**Analyzing data/Python:**

1. Analyse and process the activities dataset  
  
The API call was filtered to search for records tagged with all three surfing activities; surfing, windsurfing and kitesurfing.   
  
The data was processed as follows:  
The 'address\_region' (county) was extracted from the 'address' column using a **reusable custom function** and this was added as new column to the dataframe as it is needed for joining.  
The data was **sorted** by 'name' and 'address\_region' (county) in ascending order to prepare for removing duplicates.  
A small number of records had duplicate 'name' and 'address\_region', they only differed on the tag field, **duplicates were removed** and the last record was kept because it seemed to have more tags in general.   
The mains fields of interest were extracted using **.loc** to select 'name', 'address\_region', 'tags' columns.  
The number of rows **grouped by** 'address\_region' (county) were counted and saved in a new DataFrame df\_surf\_venue\_per\_region for analysis.

**Slicing** and **iloc** was performed on the df\_area\_type DataFrame which contained the wikipedia table to create a df\_region\_type DataFrame. This was just a lookup table to find counties ('address\_region') which contain a city.   
df\_surf\_venue\_per\_region and df\_region\_type were **merged** using a **left join** because all rows from df\_surf\_venue\_per\_region are needed. The join used columns and they had to be specified as they had different names in both DataFrames.   
The majority of counties have no cities so missing values were **replaced** with 'No City' using **fillna**.

2. Analyse and process the attractions and accommodation dataset

In this case there was no filtering applied to the attractions dataset on retrieval, the entire csv was read into a DataFrame.

The data was processed as follows (similar to above):

**Duplicate** records were dropped.  
As small number of records that didn’t have an 'AddressRegion' field populated were **dropped** as this field is needed for analysis/joining.  
Other empty cells in the other columns were filled with 'Unknown' using **fillna**.  
Only columns of interest were selected and the DataFrame was **sorted** in ascending order.  
The data was **grouped by** 'AddressRegion' to get the count per County ('AddressRegion').   
The resulting DataFrame has an **index** by default, it can be used for joining.   
The df\_county\_in\_province DataFrame is a lookup table, for each county ('address\_region') a province can be retrieved. The **index** on df\_county\_in\_province was set to to 'county' for joining efficiency.   
The tables were **Left joined** using **indexes** rather than columns. However, an inner join should work here too, as every county is in a province.

The accommodation dataset was analysed and processed similar to the attractions dataset, resulting in a dataframe containing a count of the accommodations per county (‘address\_region’).

3. Finding correlation between attractions and accommodation datasets and calculating percentage

The **Numpy** corrcoef function was used to determine a correlation between the attractions and accommodation count for Leinster.   
It showed a strong correlation: [[1. 0.97128864], [0.97128864 1.]]

Numpy was also used to calculate the percentage of the accommodation in each county. Using a numpy array is useful here as there is no need to iterate though the array to do the calculation on each item.

A **dictionary** was populated by iterating through the list of percentages and populating the key value pairs with the county and percentage, see output:  
  
{

'Carlow': 1.0,

'Cavan': 2.0,

'Clare': 7.0,

'Cork': 11.0,

'Donegal': 8.0,

'Dublin': 9.0,

'Galway': 10.0,

'Kerry': 13.0,

'Kildare': 2.0,

'Kilkenny': 2.0,

'Laois': 1.0,

'Leitrim': 1.0,

'Limerick': 2.0,

'Longford': 0.0,

'Louth': 2.0,

'Mayo': 6.0,

'Meath': 2.0,

'Monaghan': 1.0,

'Offaly': 1.0,

'Roscommon': 1.0,

'Sligo': 3.0,

'Tipperary': 3.0,

'Waterford': 3.0,

'Westmeath': 1.0,

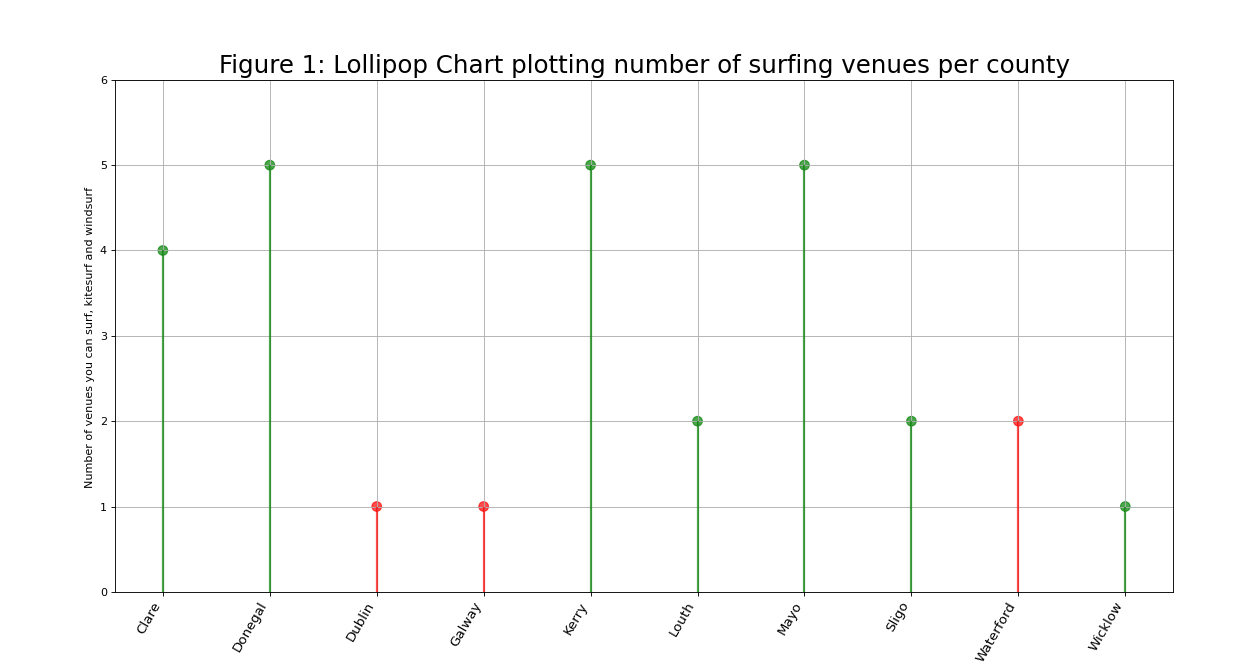
'Wexford': 3.0,

'Wicklow': 3.0

}

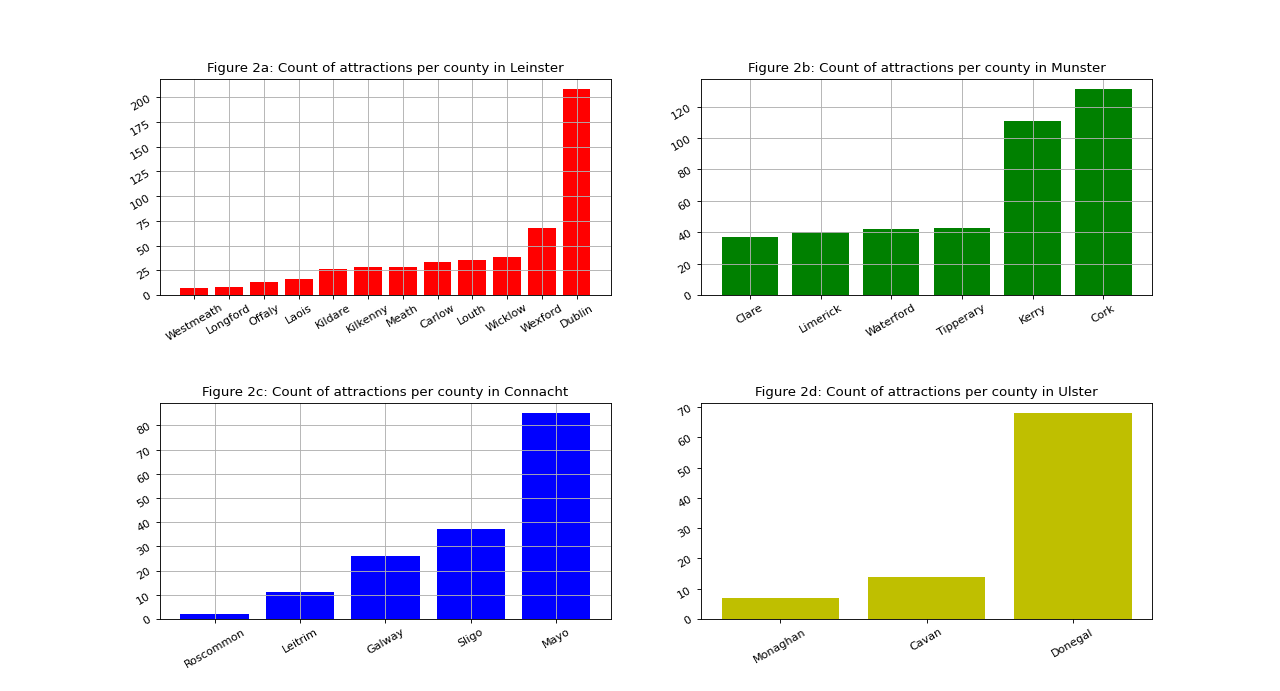
**Visualizations/Insights**

Below is the **lollipop chart generated using matplotlib**, which displays the categorical data, similar to a bar chart but a bit visually cooler. It shows the total number of venues which cater for multiple types of surfing within a county in Ireland. The lollipops in RED show counties that contain a city, those in green do not. The colour is set based on a DataFrame column, the function to do this uses a **list comprehension to iterate** over the values in the column.

**Insights:**

* Donegal, Kerry and Mayo have the most venues to choose from.
* Counties with cities (seen in red in the plot) have very little options for surfing.

Below is the grid of subplots generated using the matplotlib plt.subplots function. The data in the aggregated attractions DataFrame was separated out based on province and used for each of the bar plots below. This was a good type of chart to visualise the data for the four counties separately but at same time.

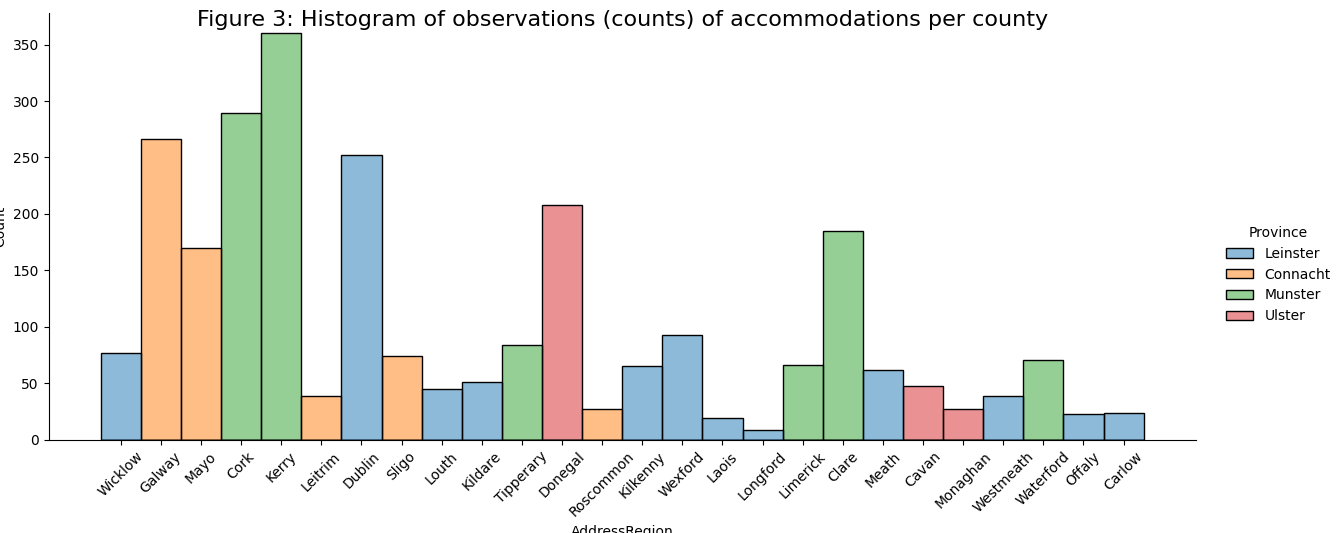


Insights:

* Dublin has by far the most attractions in Leinster, Donegal in Ulster and Sligo in Connacht.
* In Munster Cork is ahead but Kerry is very close. Munster is the only province with two strong counties.

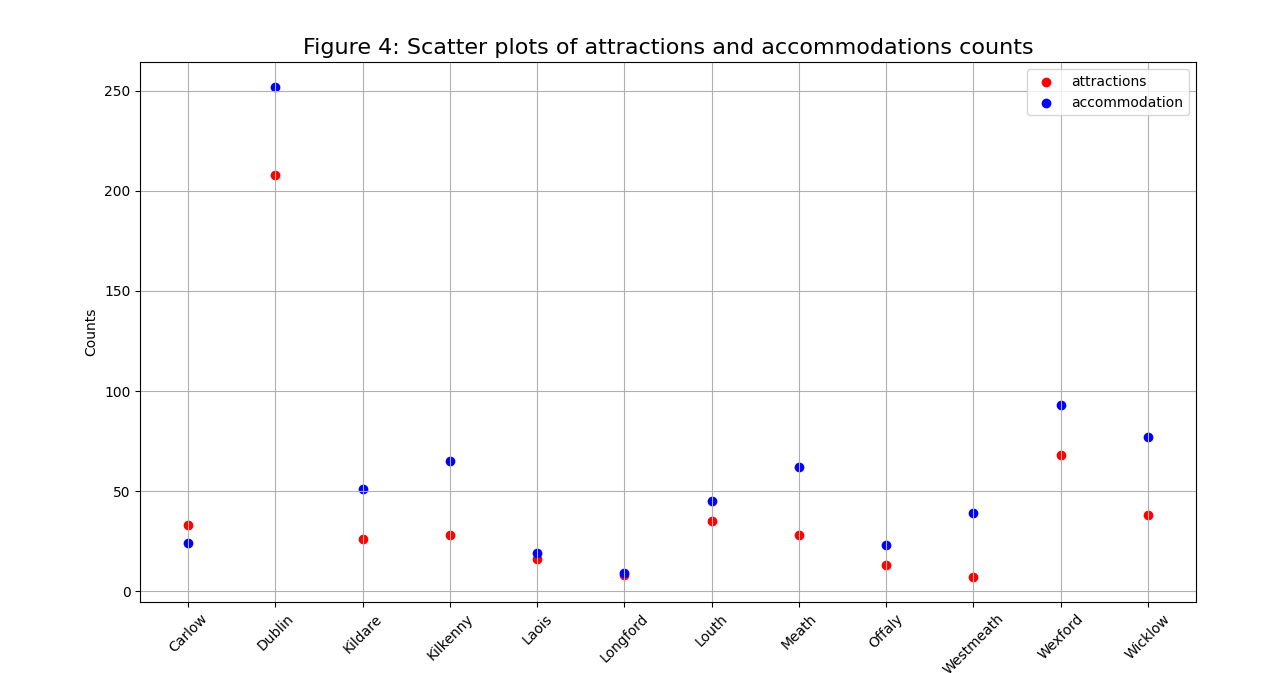
# Below is a histogram created using the Seaborn distplot() function. It shows the number of observations per county in the attractions dataset (this is the data before it is aggregated unlike the previous examples. Here Seaborn does the counting). The hue parameter is set to distinguish between provinces.

# 



**Insights:**

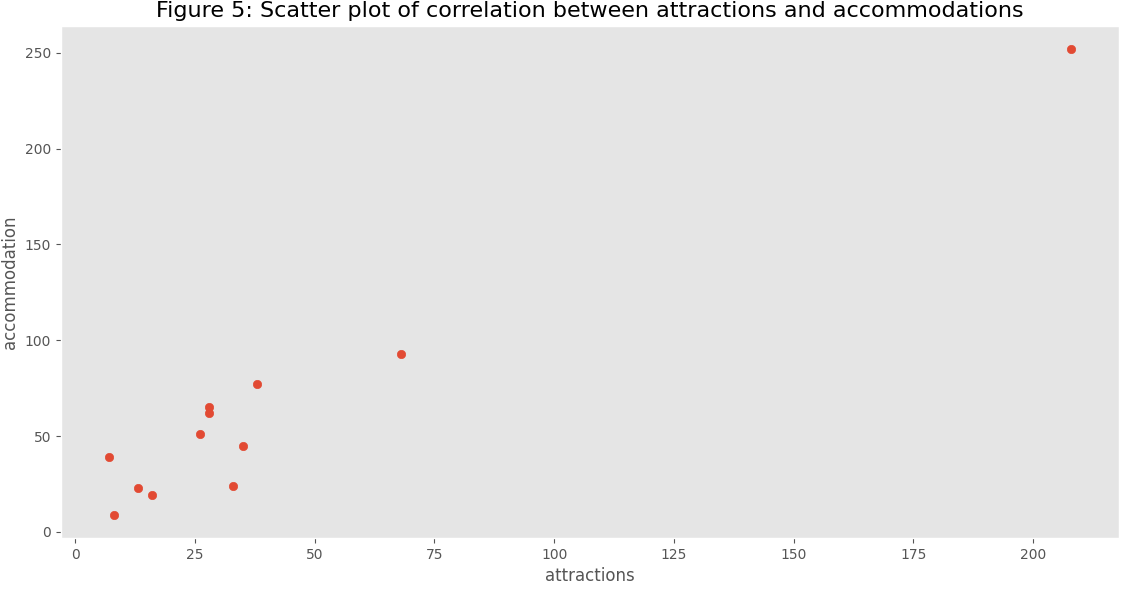
* Kerry has more accommodation listed than any other county.
* The two counties with the highest counts are in Munster.
* Surprisingly Dublin only comes in fourth after Galway.
* Longford has the least amount of accommodation.
* In Leinster there is a big difference between counts for Dublin and the other 11 counties.

Below is a plot containing two scatter plots in the one plot created using matplotlib. The objective here was to see if there was any correlation between the attractions per count and the accommodations per county in Leinster.  
  


**Insights:**

* Appears to be correlation between the count of attractions and count of accommodations per county.
* This may not be the best plot to visualize a correlation, need do investigate further to be sure of correlation (see next graph).

Below is a plot of the high positive correlation between the number of attractions and the number of accommodations in each county in Leinster.



**Insights:**

* There is strong positive correlation (0.97128864) between the accommodation and attractions count.
* Within Leinster both the accommodation and the attractions are far great in Dublin than any other county.