DATA9005: Data Visualisation

Dublin Ride Bike Data and Met Eireann Weather Data

Brian Higgins - R00239570

1. Introduction

This project will look at visualisation with Python and will explore the following areas:

- Exploring large datasets
- Combine different datasets on Time Series
- Python Data Cleaning with Jupyter Notebooks
- Visualisation Techniques with Tufte's Principles
- Matplotlib Library
- Seaborn Library
- GeoPandas Maps
- Plotly Dashboard

1.1 Project idea

I was interested in working with large amounts of Time Series data and the data I had for the Dublin Bikes was over 3.43GB with 32 Million observations. I also wanted to combine this Time Series data with weather data from Met Eireann, which was in a different time range. A third aim was to use Geospatial data for the bike stations.

One flaw in the data was that the Dublin Bike data did not include "customer", or "length of trips", etc which is included in Cork Bike data for example, as this data is not made available publicly. This added a challenge to explore the data for many of the plots.

2. Data

2.1 Data Source

Dublin Bike Ride Share Data: Consisted of 12 csv files which were joined together to create 3.43 Gb of data with over 32 million observations from 2018 to 2019 with several observations taken for each hour.

Data Source: https://data.gov.ie/dataset/dublinbikes-api

Met Eireann: The second data source was for weather from 2006 to 2023.

Data Source: https://www.met.ie/climate/available-data/historical-data

2.2 Large Datasets and Combing datasets

Exploring data with 32 million observations was a challenge, especially for my laptop. I created a function to reduce the number of observations to an average for each hour. Each bike station was entered to the function, resampled and then added to a new dataframe. This reduces the data down to 3 million rows. I have also created several smaller dataframes to explore the data.

- Total data: cleaned data with 3 million observations
- Selected_stations_df: 5 bike stations
- Total_data_point_station / total_data_smithfield_station
- Stations_explore: Geospatial data

3. EDA and Data Cleaning

3.1 Jupyter Notebook

I created one Jupyter Notebook, but in the future I would break this up. I have commented out section 4 as this is processing data on 3.43GB of data. I have saved a copy of this process as a PDF so you can see it working (although the Plotly plots do not save to the pdf).

3.2 Data Cleaning

In section 4 (contained in the PDF) there is EDA and numerous data methods including:

- Exploring Missing data
- Duplicate data
- Changing data types
- Combing data on time series
- Dropping columns
- Feature Creation with Time Series
- Feature Creation on categories and numeric columns

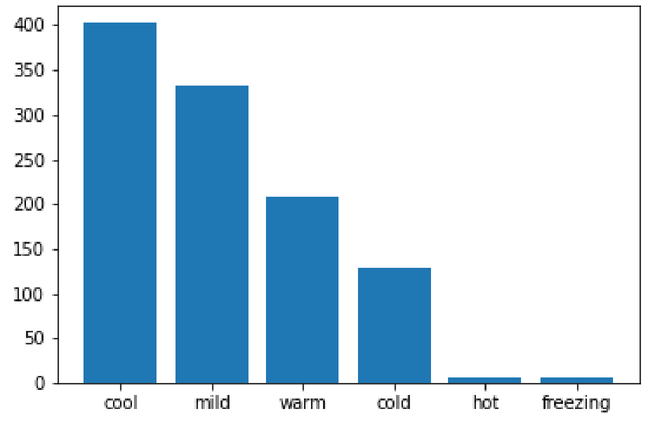
4. Visualisation Theory

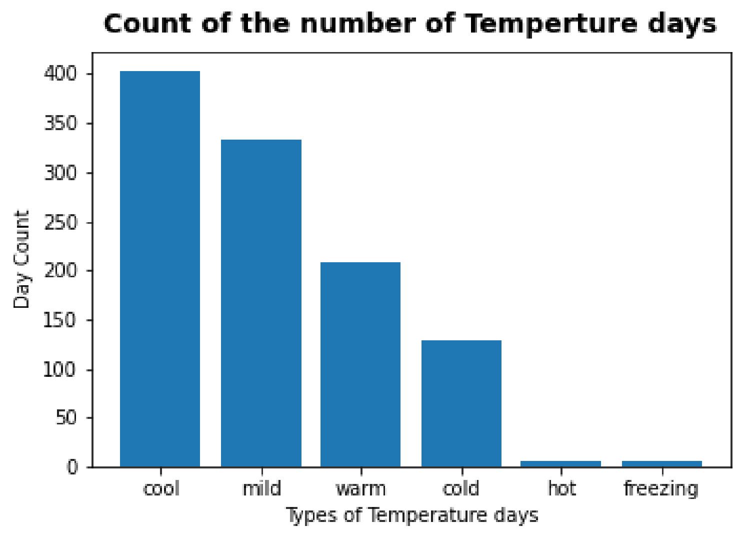
Edward Tufte is a data visualisation expert and professor of political science, statistics and computer science at Yale University. Tufte set out a set of principles for effective presentation of information. While Tufte set out 6 principles these can be explained in a few more points for clarity.

1) **Show the data:** Present data in an honest way.
2) **Maximise the data-ink ratio:** Minimize the use of non-data ink such as grids, borders and over junk that distracts from the data.
3) **Use small multiples plots:** Use multiple variations of the same chart to enable comparisons.
4) **Use clear labels:** Label all parts of a plot.
5) **Highlight Important information:** Emphasis important parts with colour, labels, fonts, etc.
6) **Avoid junk**: However, do not use any distracting elements.
7) **Use appropriate charts**: Choose the right chart for the right data
8) **Provide context**: Show the data in a meaningful context

This list expands on Tufte's list of principles clearly. Throughout the Jupyter Notebook I have used these principles to look at the data. At times I have not fixed issues to show comparisons. Below we will see some examples where things can be improved.

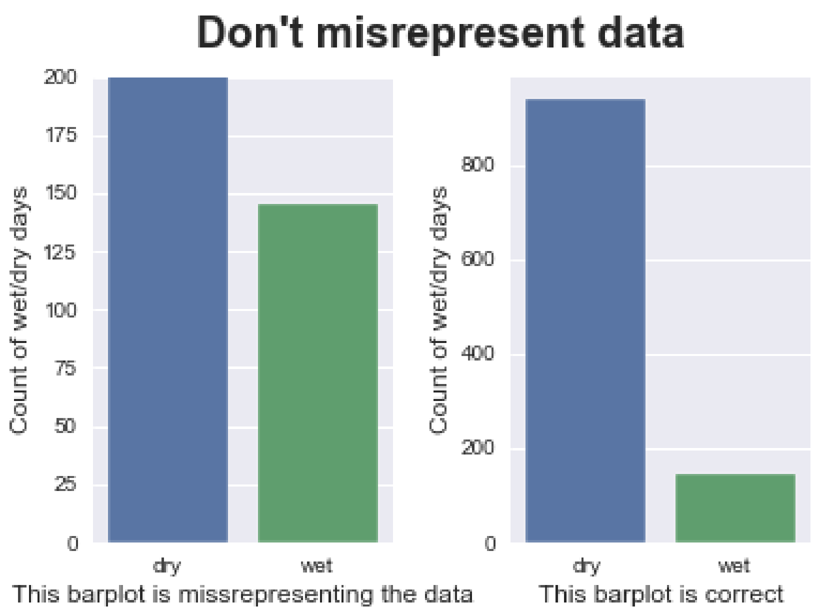
4.1 Make clear labels





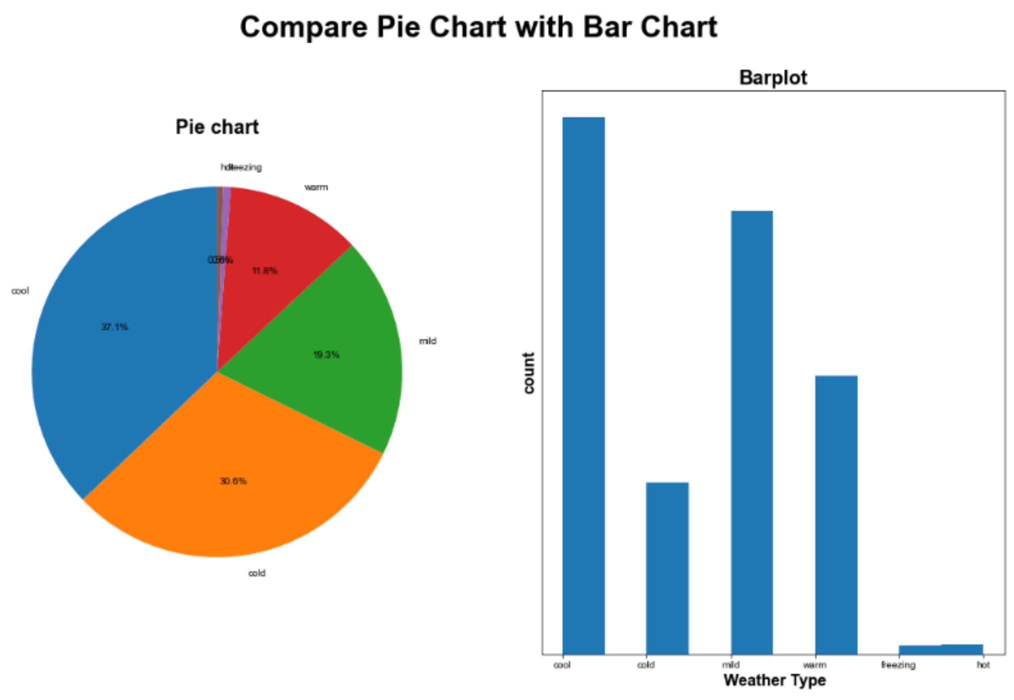
The plot on the left has x and y values but no labels telling us what they are. The plot on the right improves things by giving the x and y-axis labels and including a title.

4.2 Don't misrepresent the data



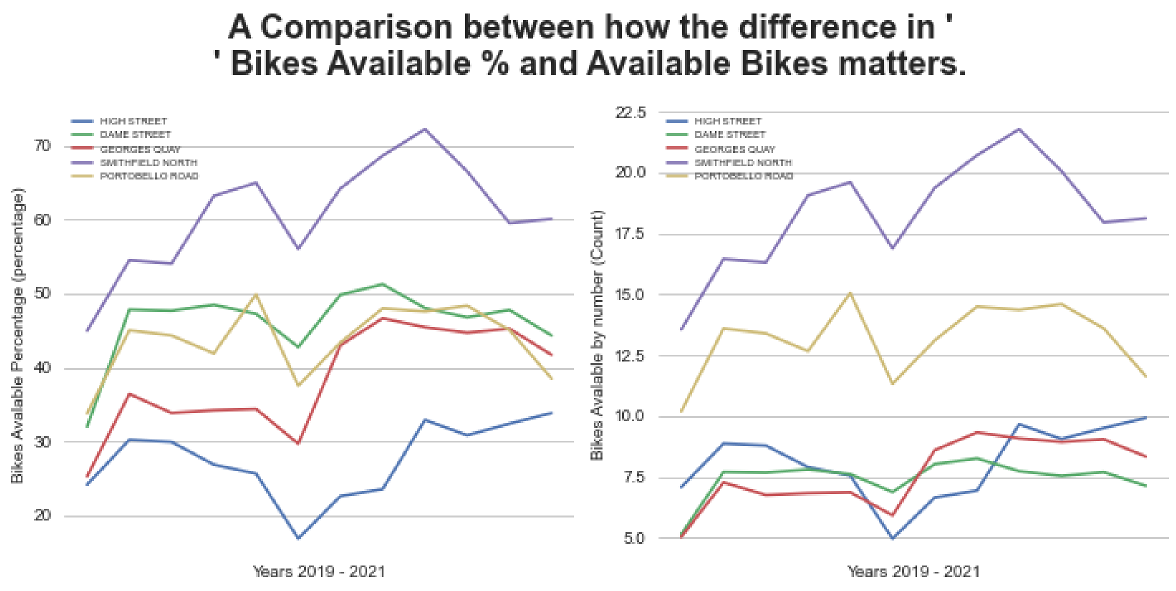
Here we have the same data on two bar plots. However, because the y-axis on the left plot only goes to 200, it is misrepresenting the wet days. While the wet day number is the same in both plots which is not the case.

4.3 Use appropriate charts



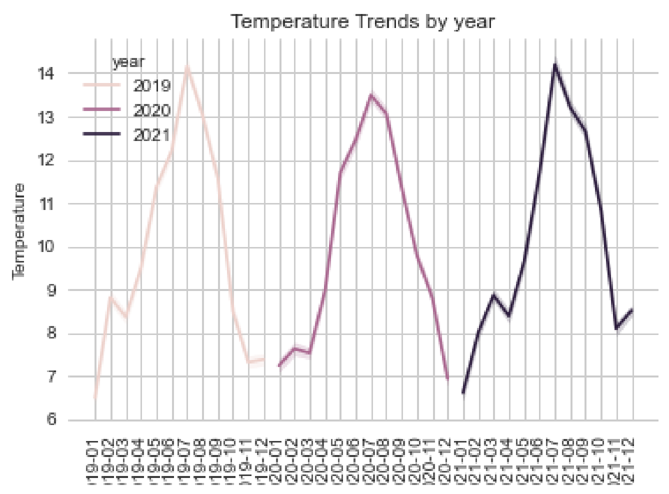
I have left issues with the Pie Chart to show how it can be messy. While the barplot looks less interesting it gives a clearer representation of the data (I can, of course, improve on this with colours but left it bare for a reason). The Pie Chart needs percentages to be able to read the proportions as the human eye finds it hard to read angles and notice now the smaller values are harder to read and the labels are squashed.

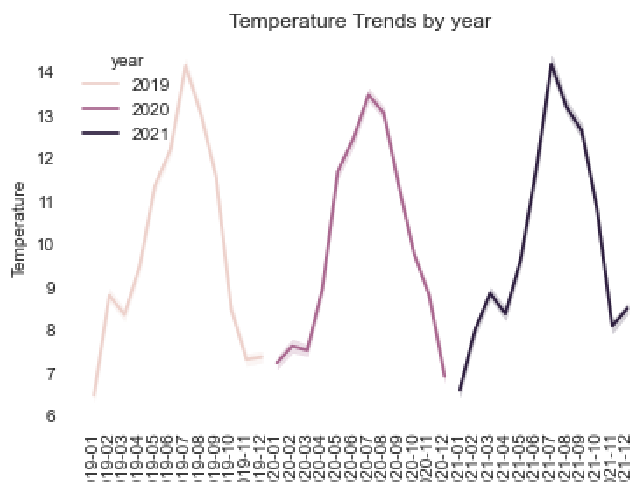
4.4 Provide Context



Here we have two near identical plots with the availability of bikes. However, the number of bikes available at each station varies and it goes up to 40. So a percentage of availability is better and notice how the stations lines then change in the second plot.

4.5 Avoid Junk



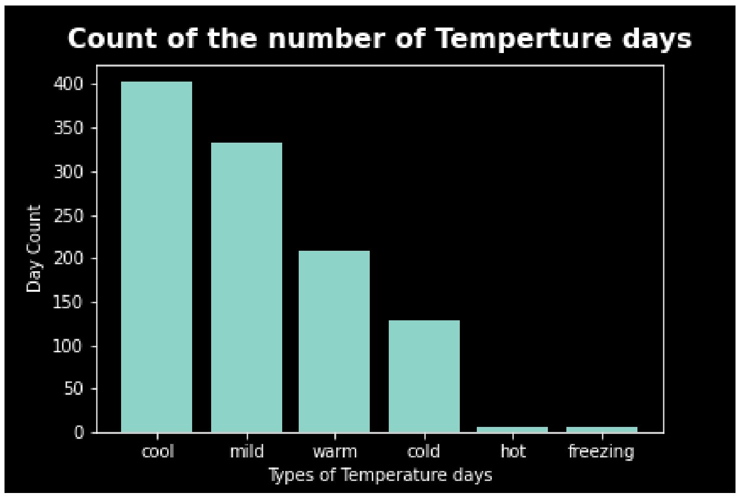


Here we have the average temperatures for three years, the grid just adds noise to the plot and is not needed.

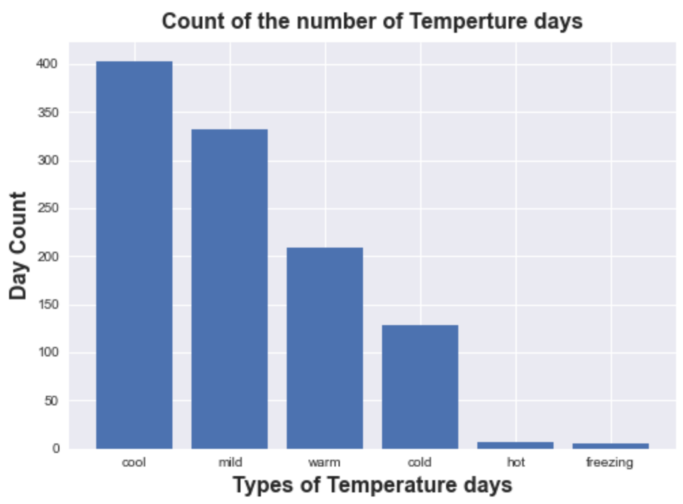
5. Matplotlib Plots

Matplotlib is a Python Library used for data visualisation, it has a wide variety of functions and tools for creating different types of charts, plots and graphs. Matplotlib was originally created in 2003 and is still an important base package. Matplotlib provides a high level of customization for visualisation. In the Jupyter Notebook, I explore various plots in detail.

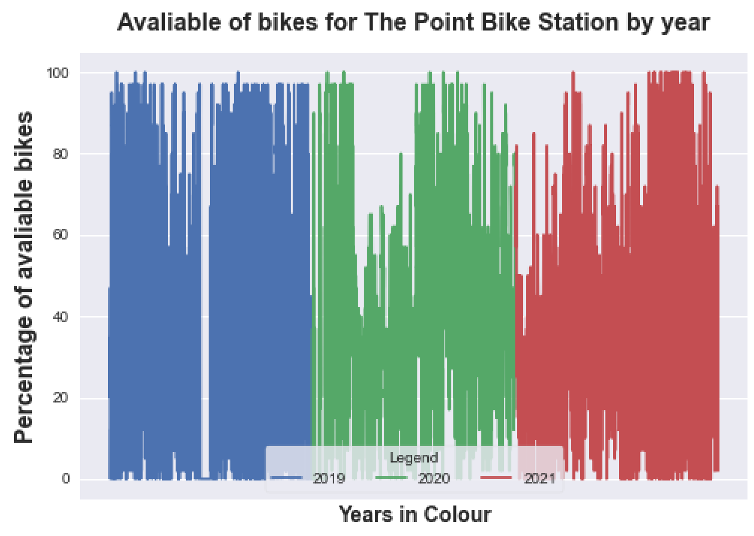
<u>Bar Plots: Different Styles</u>



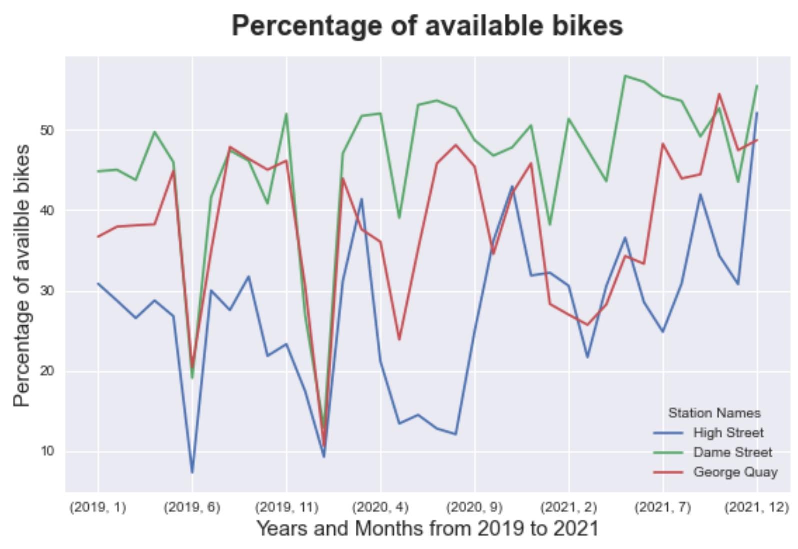
<u>Bar Plots: Different Styles</u>



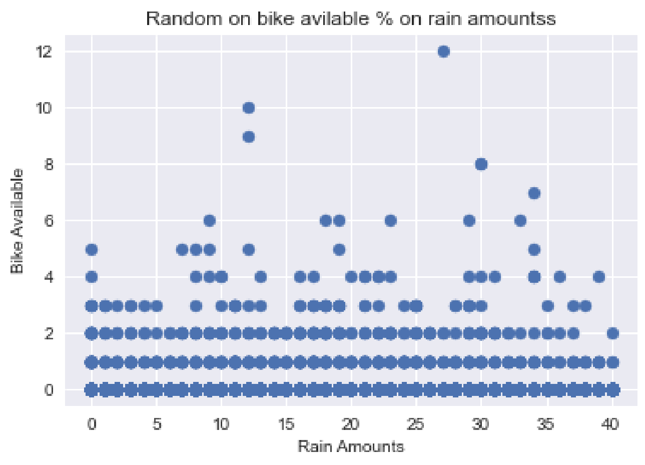
Line plots: By day



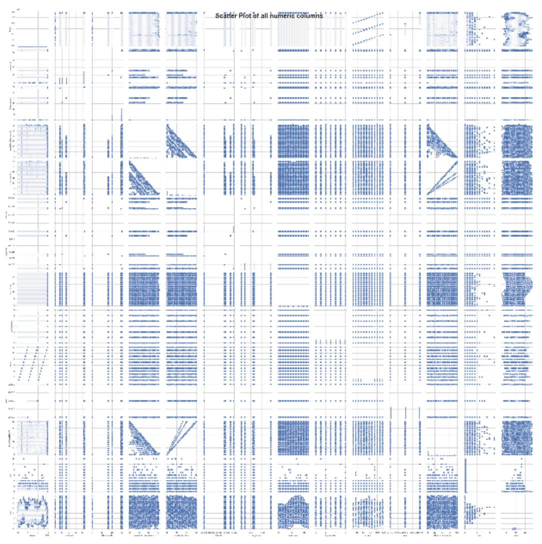
Line plots: Multiple stations



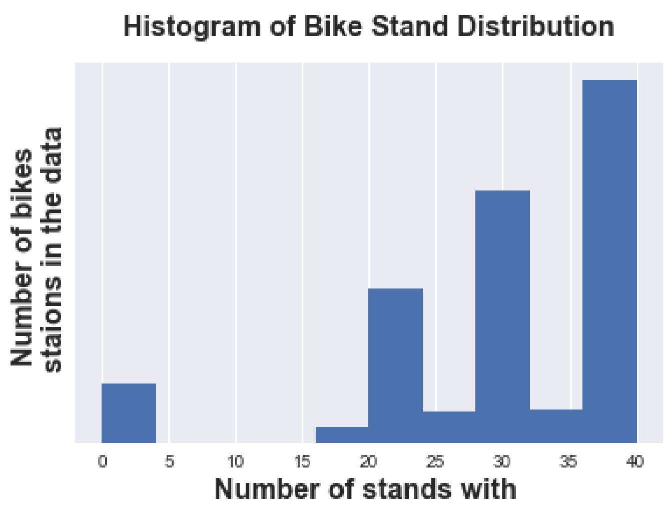
Scatter Plots



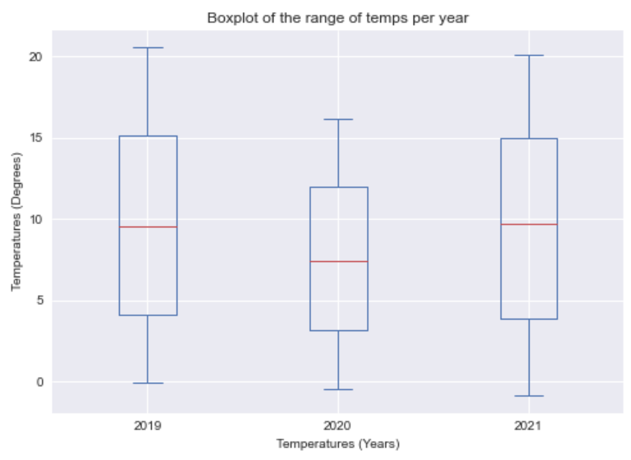
Pairwise Plot: Useful for EDA



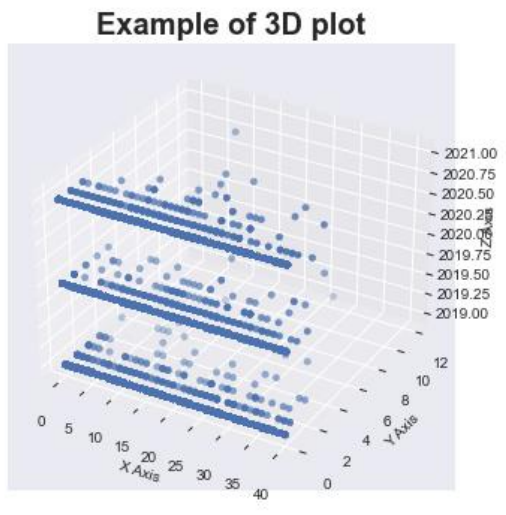
Histogram



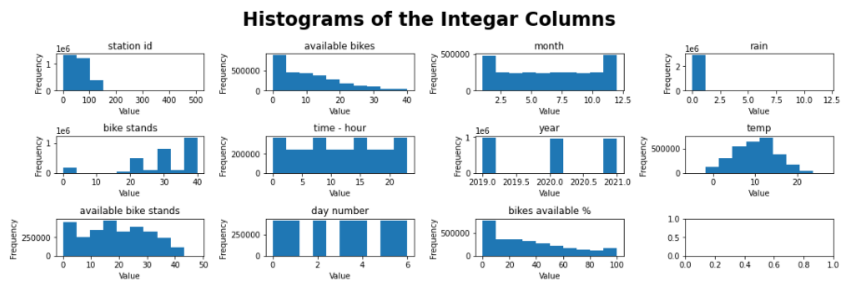
Boxplot



<u>3D Plots: Not useful</u>



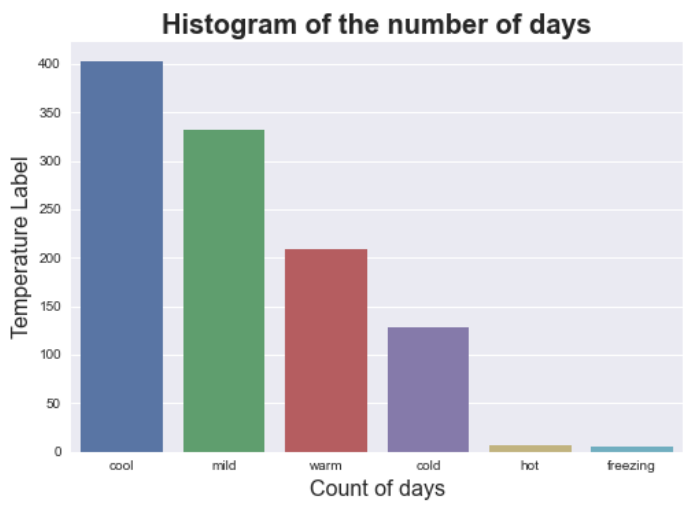
<u>Multiple Plots: Showing Distribution</u>



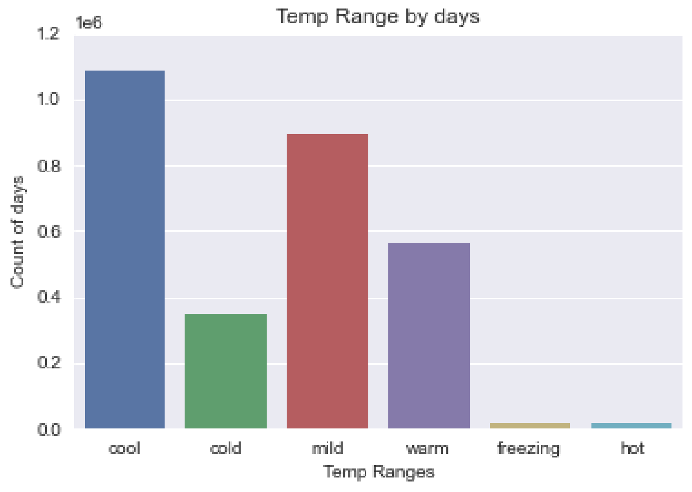
6. Seaborn plots

Seaborn is based on Matplotlib and builds on top of Matplotlib with a range of visualisations including distribution plots, regression plots, etc.

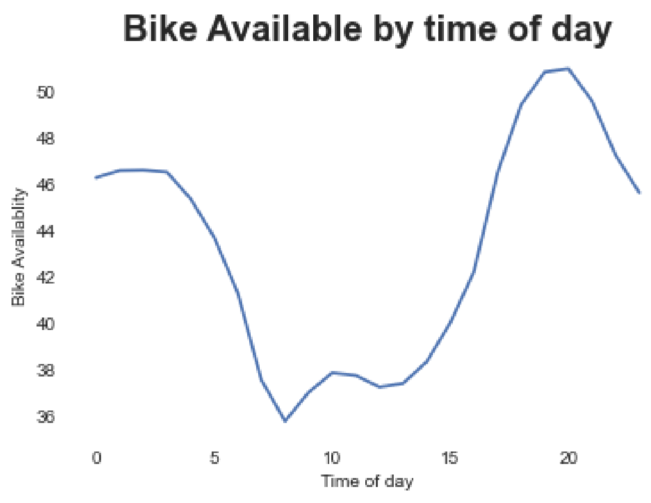
<u>Histograms</u>



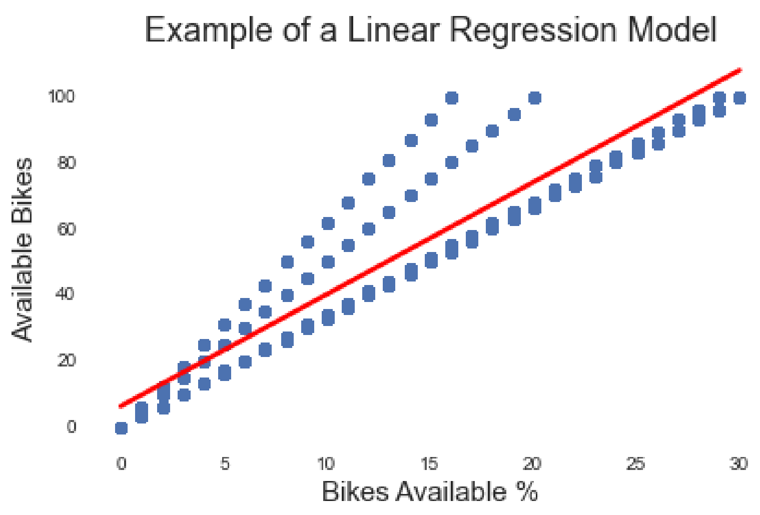
<u>Count Plots: Counts all observations (Notice y-axis values)</u>



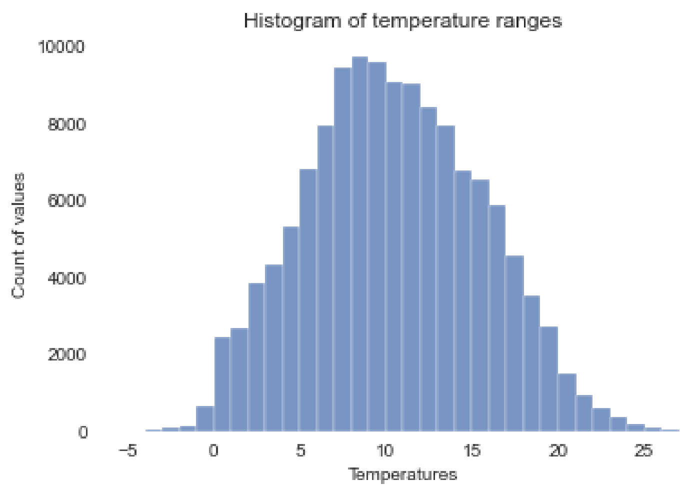
Line Plot



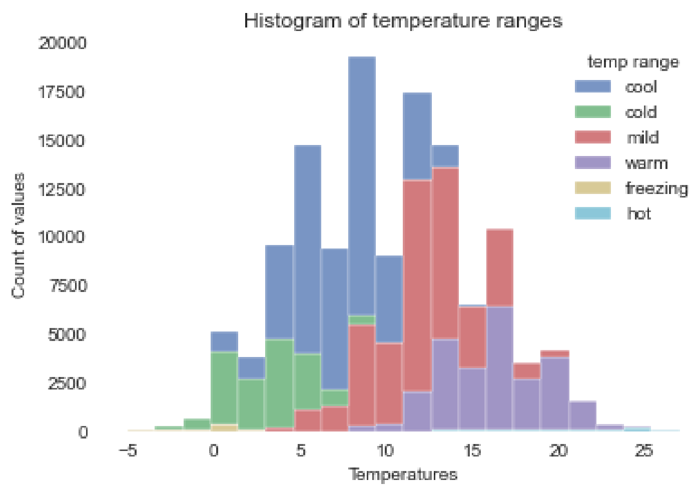
Line Plot with Regression Line



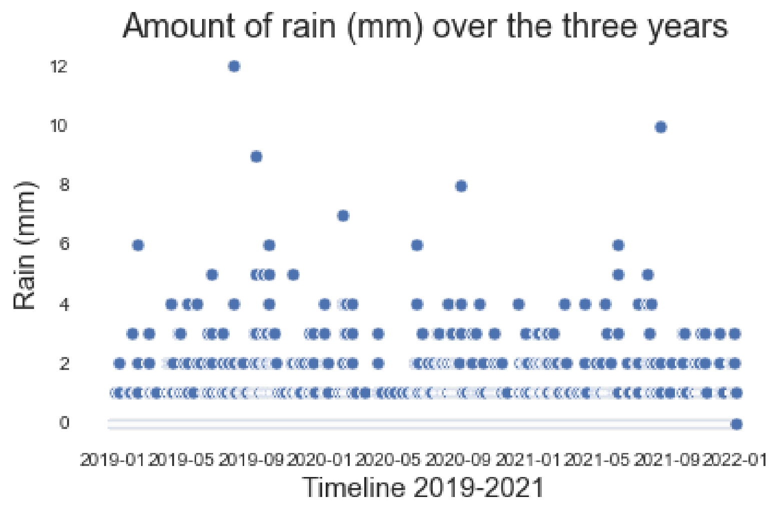
Histogram: Basic



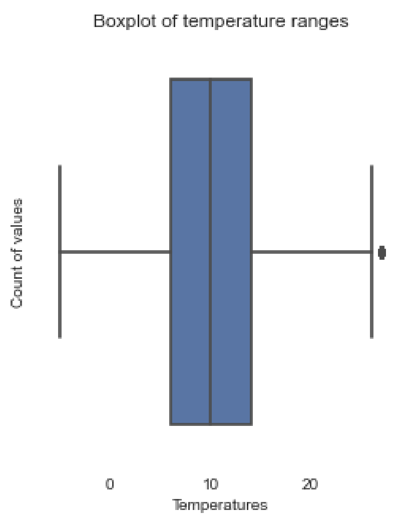
Histogram: Group



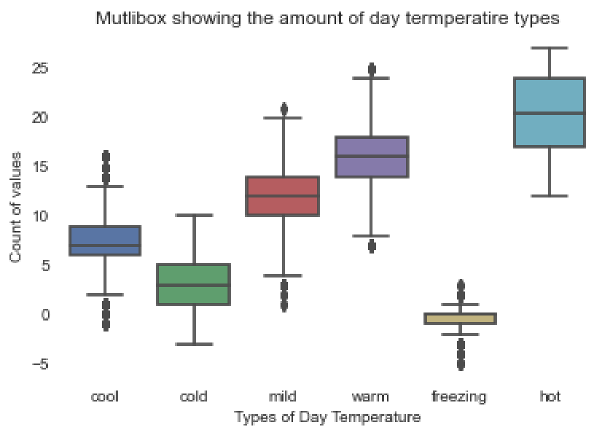
Scatter Plot



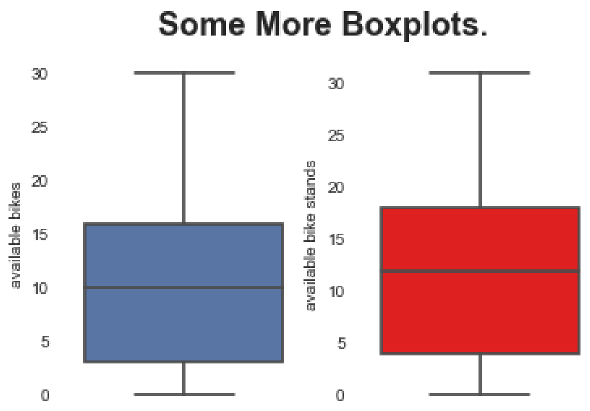
Boxplot: Basic



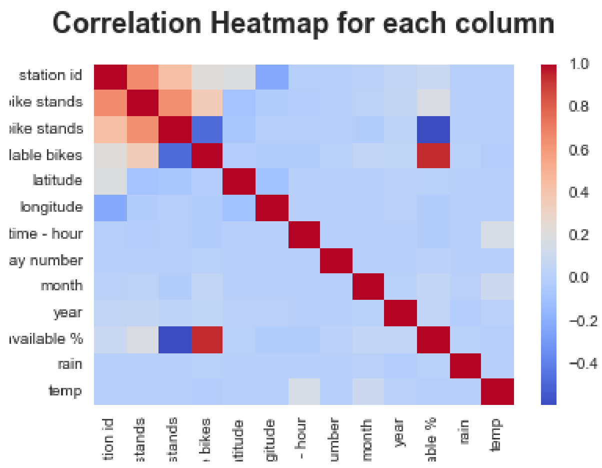
Boxplot: Group by column



Boxplot: Multiple Plots



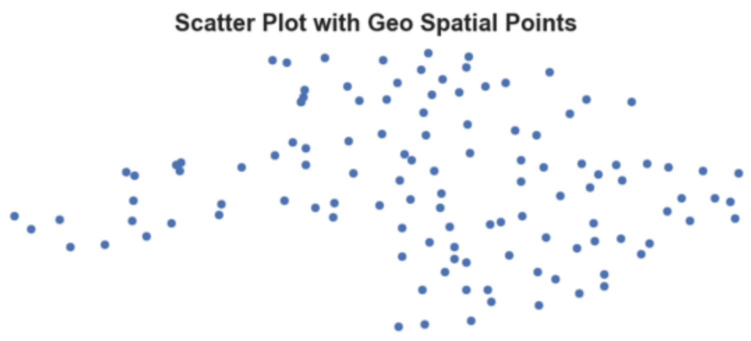
Heat Map



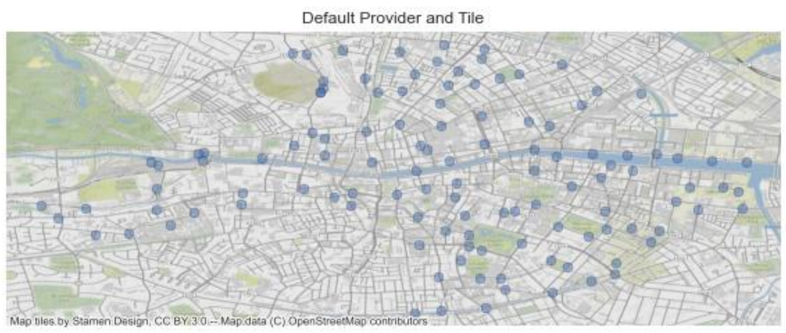
7. Geospatial Maps

Geospatial maps are maps that represent geographic locations and I have used the GeoPandas library. You use Latitude and Longitude to create Geometry points that you can then map. You build up layers of maps with CRS (Coordinate Reference System) so that each layer can be matched together. You can also create Polygon to group points into areas.

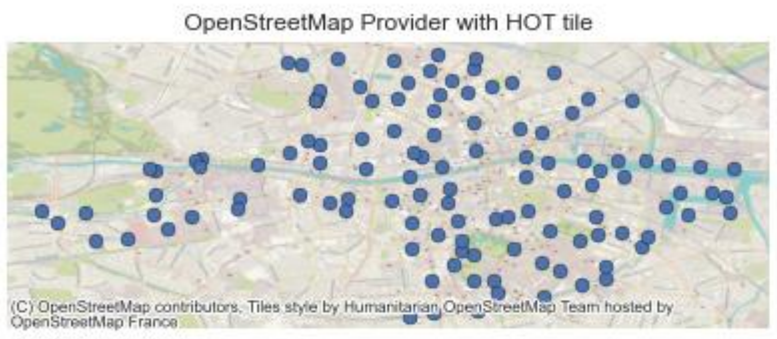
GeoSpatial: Geometry Points: 1 Layer

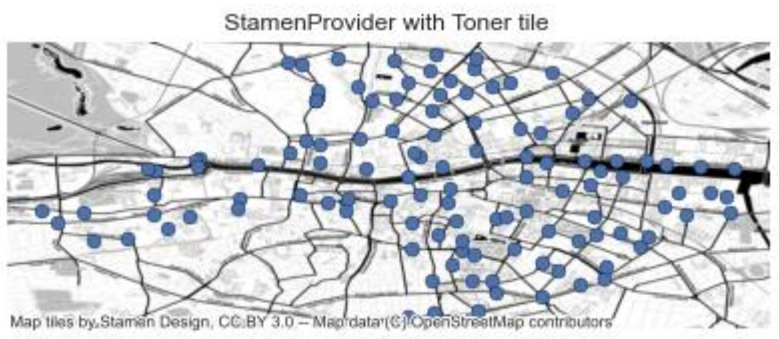


GeoSpatial: Geometry Points and base layer

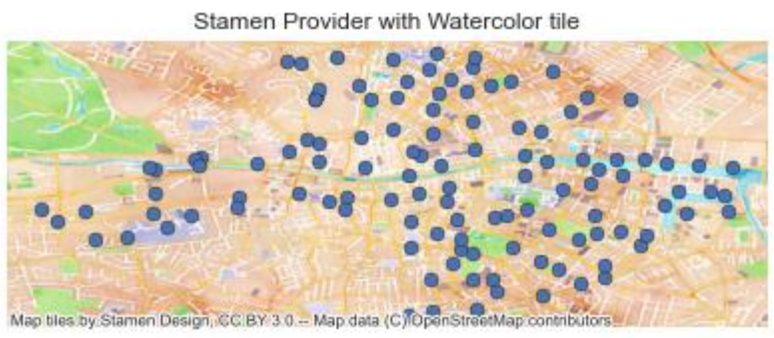

Default Provider and Tile
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

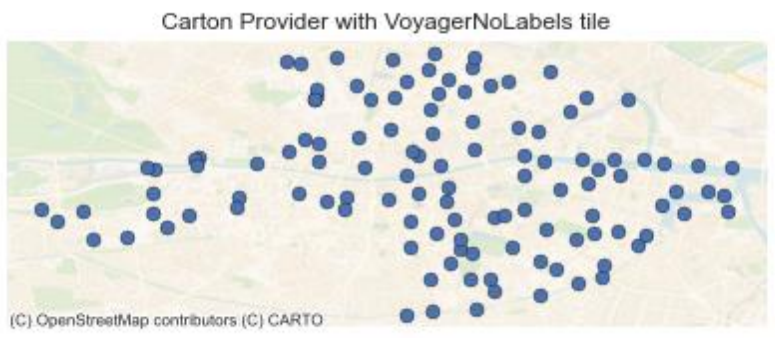
More base maps with different base maps.


OpenStreetMap Provider with HOT tile
(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap Team hosted by OpenStreetMap France

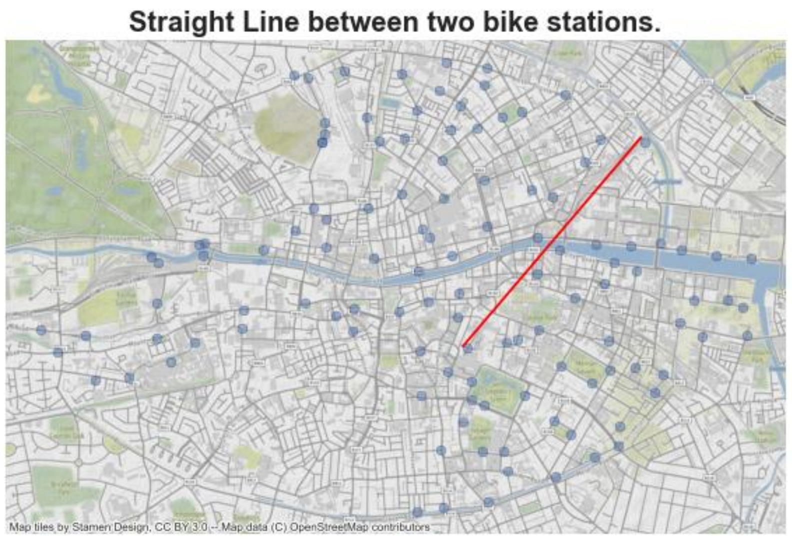

StamenProvider with Toner tile
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

More base maps

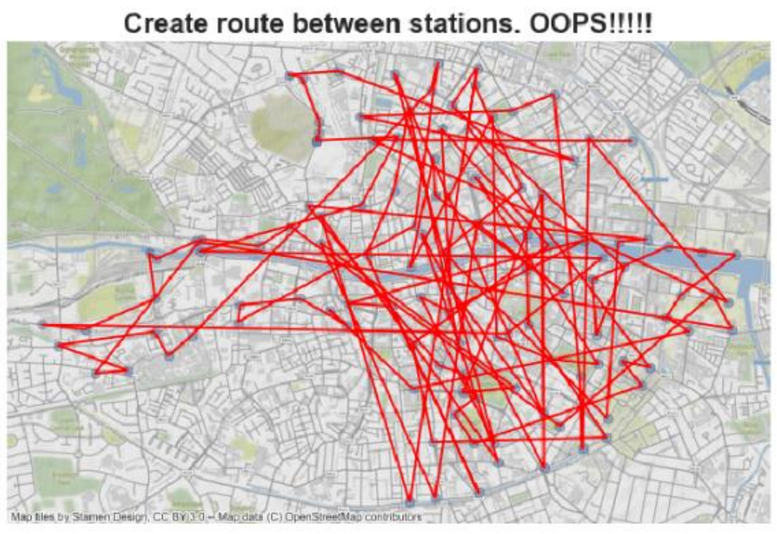

Stamen Provider with Watercolor tile
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors


Carton Provider with VoyagerNoLabels tile
(C) OpenStreetMap contributors (C) CARTO

Route 1: Straight Line

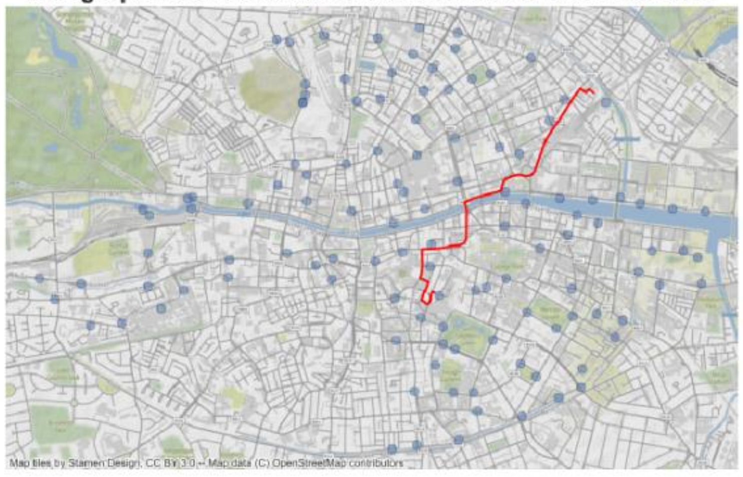

Straight Line between two bike stations.
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

Route 2: Go through each station in between


Create route between stations. OOPS!!!!!
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

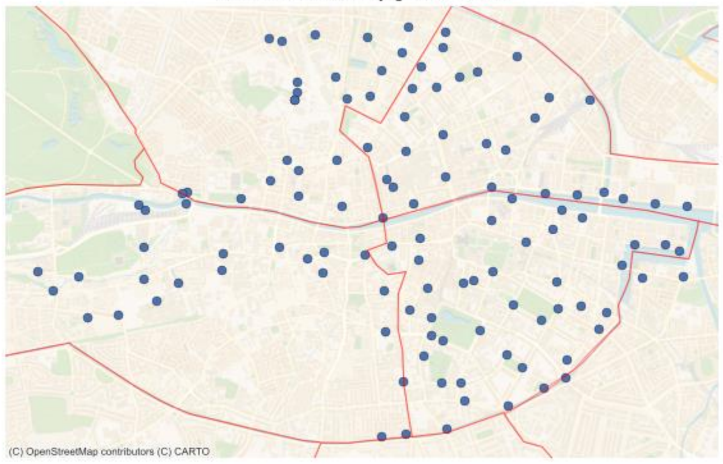
Route 3: Use OpenRouteServices with their API

Using OpenRouteServices to create a route between stations


Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

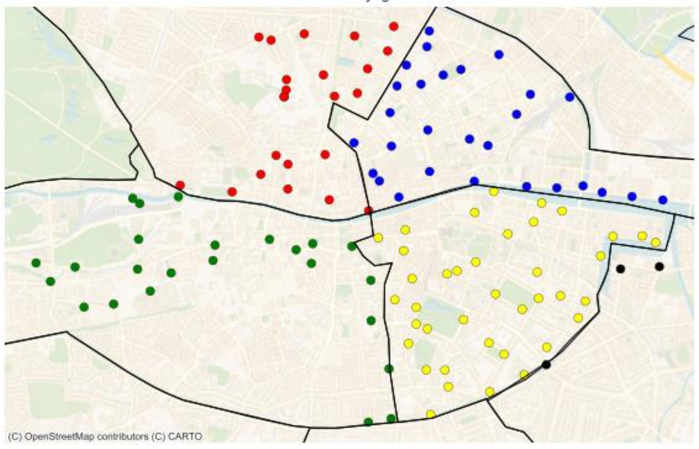
Use Ploygons for Dublin Postal codes with shapefiles

Carton Provider with VoyagerNoLabels tile



Use Ploygons for Dublin Postal to change colours

Carton Provider with VoyagerNoLabels tile

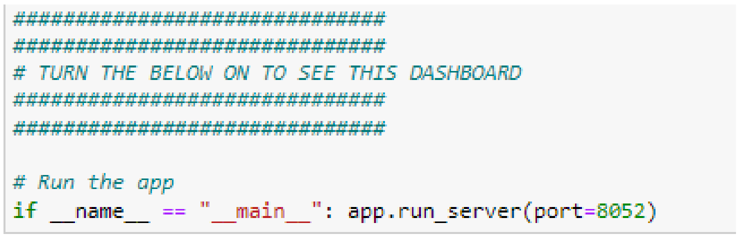


8. Plotly Dashboard

Dashboards allow us to combine many different features:

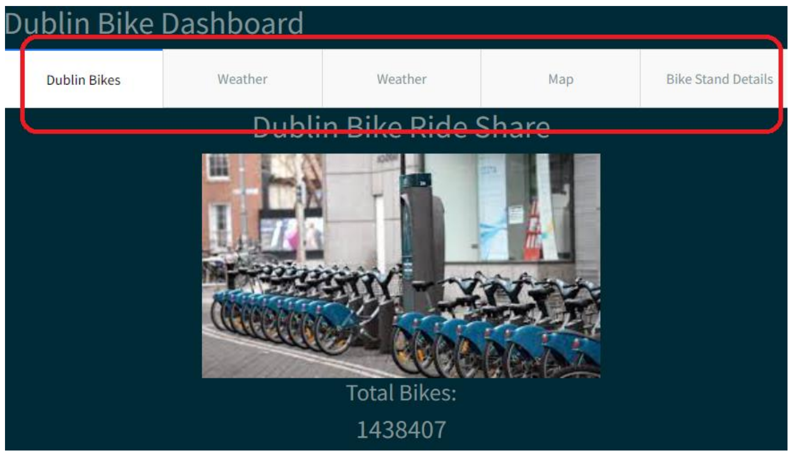
- Bar plots, Scatter Plots, Box plots, Histograms, etc
- Multiple Tabs for different information
- Adding drop-down boxes, and click boxes to add other features
- Maps and other features

Note: The last line of each Dashboard has been commented out, otherwise the notebook would stick at that point.



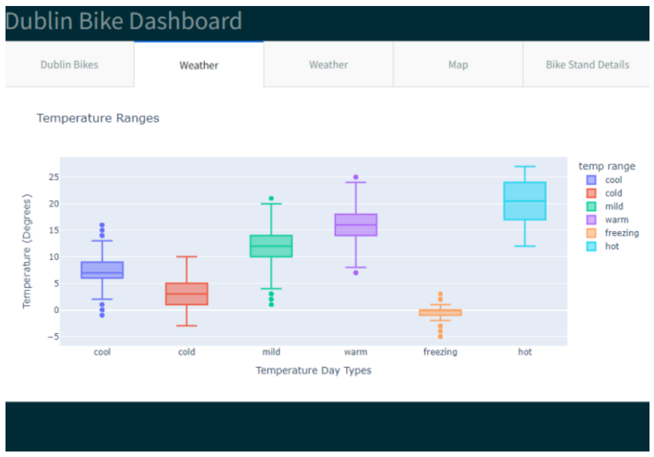
```
##############################
##############################
# TURN THE BELOW ON TO SEE THIS DASHBOARD
##############################
##############################

# Run the app
if __name__ == "__main__": app.run_server(port=8052)
```

Dashboard: Multiple Tabs

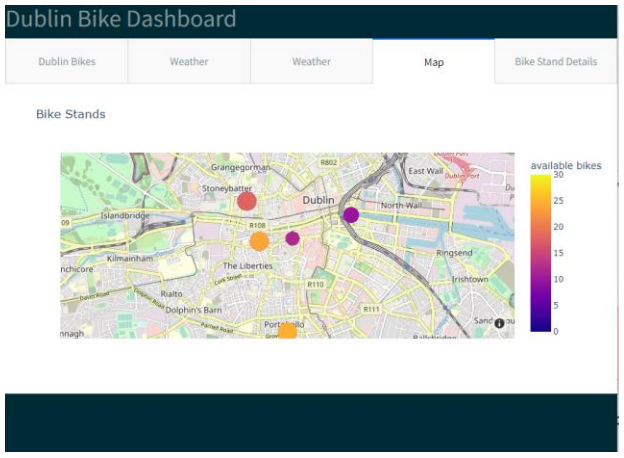


Total Bikes:
1438407

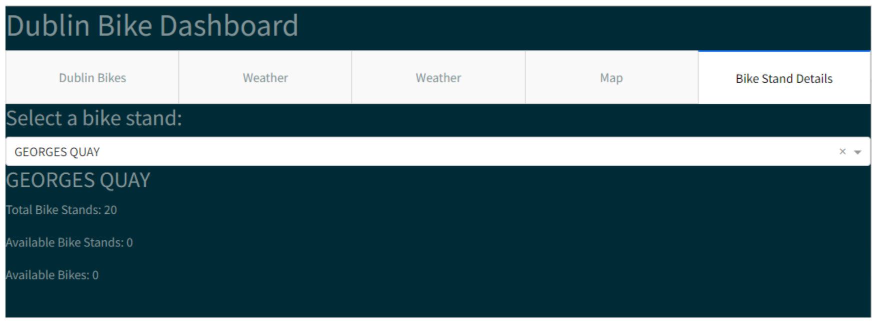
Dashboard: Different Plots



Dashboard: Maps



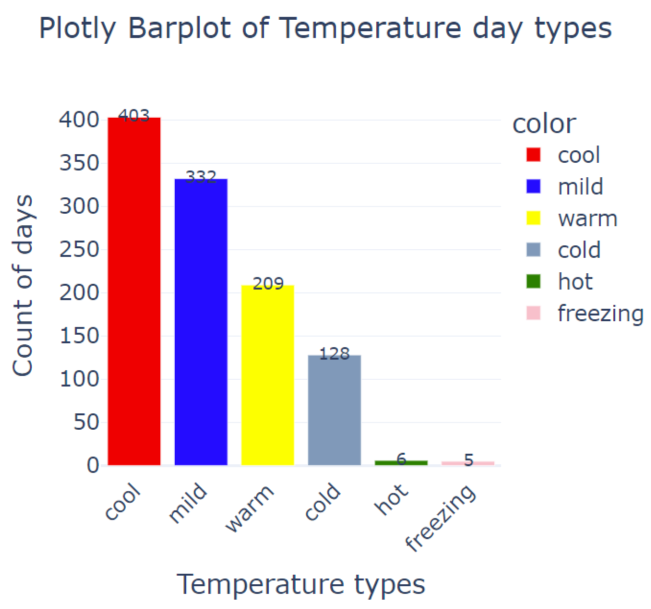
Dashboard: Dropdown Menus



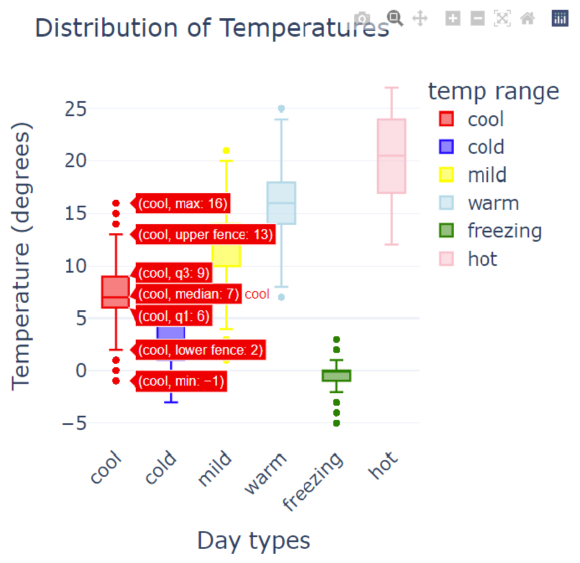
Select a bike stand:
GEORGES QUAY
GEORGES QUAY
Total Bike Stands: 20
Available Bike Stands: 0
Available Bikes: 0

8.1 Plotly Static Plots – Using some colour

Bar Plot – Colours too bright

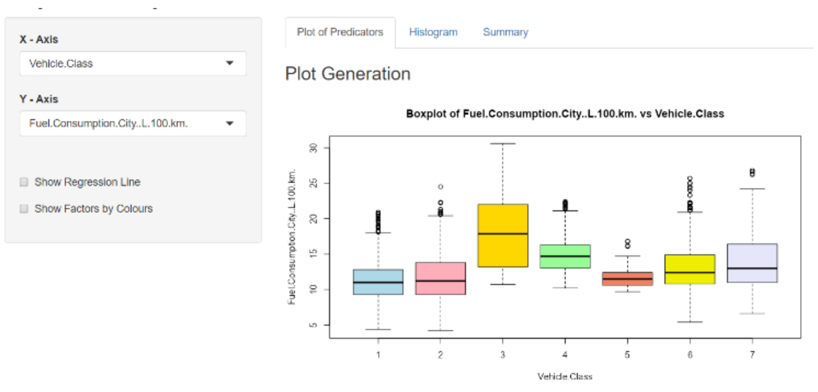


Box Plot – Notice built-in interactive features



9. R Shiny Dashboard

Shiny is another dashboard I created in R-Studio. Here you can see the same features mentioned above.

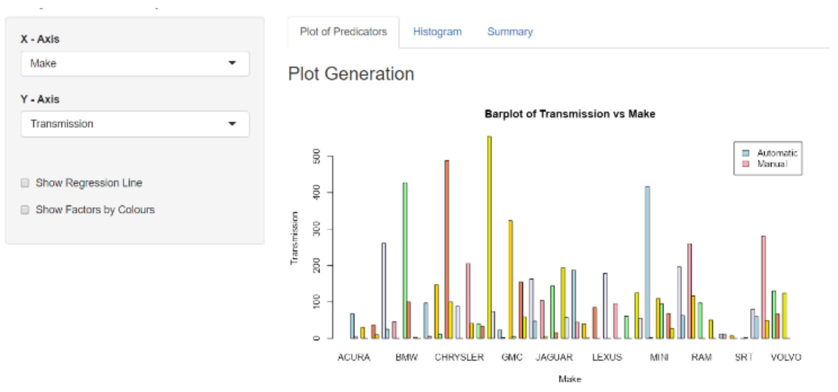


X - Axis
Vehicle.Class

Y - Axis
Fuel.Consumption.City..L.100.km.

☐ Show Regression Line
☐ Show Factors by Colours

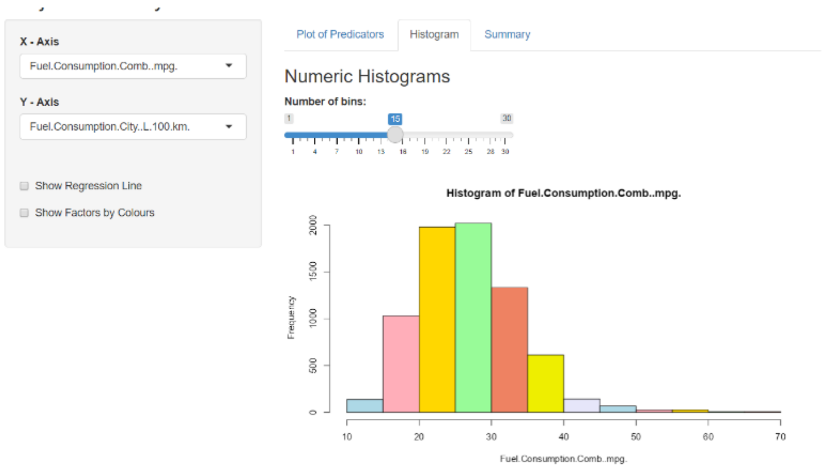
Plot of Predicators
Histogram
Summary
Plot Generation
Boxplot of Fuel.Consumption.City..L.100.km. vs Vehicle.Class
Fuel.Consumption.City..L.100.km.
Vehicle.Class



X - Axis
Make

Y - Axis
Transmission

☐ Show Regression Line
☐ Show Factors by Colours



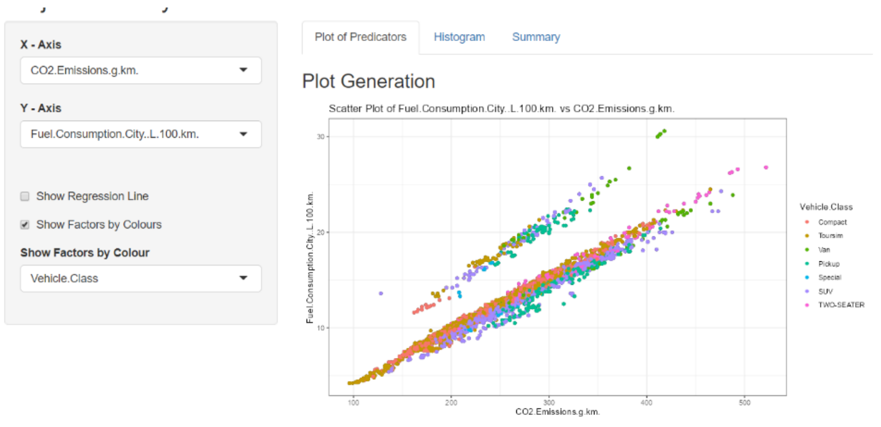
X - Axis
Fuel.Consumption.Comb..mpg.

Y - Axis
Fuel.Consumption.City..L.100.km.

☐ Show Regression Line
☐ Show Factors by Colours

Number of bins: 10

Plot of Predicators
Histogram
Summary
Numeric Histograms
Histogram of Fuel.Consumption.Comb..mpg.
Frequency
Fuel.Consumption.Comb..mpg.

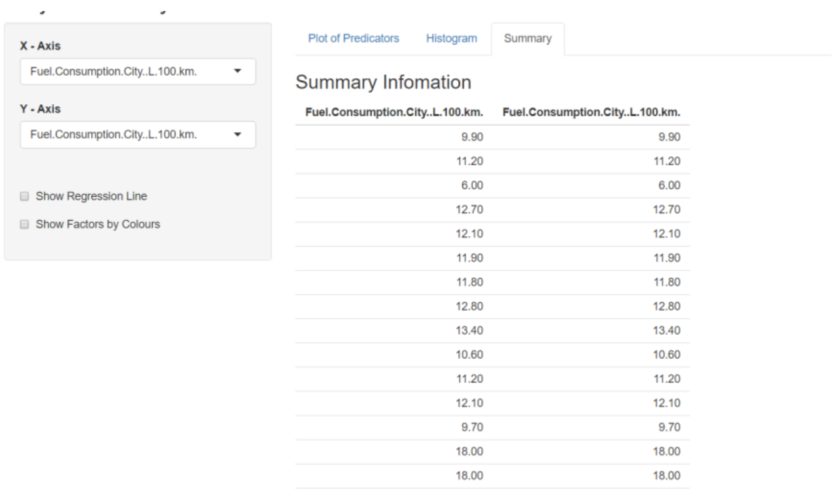


X - Axis
CO2.Emissions.g.km.

Y - Axis
Fuel.Consumption.City..L.100.km.

☐ Show Regression Line
☑ Show Factors by Colours

Show Factors by Colour
Vehicle.Class



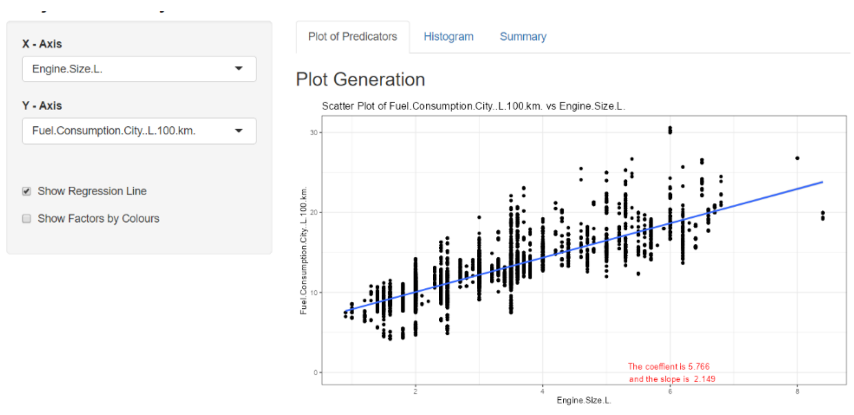
X - Axis
Fuel.Consumption.City..L.100.km.

Y - Axis
Fuel.Consumption.City..L.100.km.

☐ Show Regression Line
☐ Show Factors by Colours

Summary Infomation

Fuel.Consumption.City..L.100.km.	Fuel.Consumption.City..L.100.km.
9.90	9.90
11.20	11.20
6.00	6.00
12.70	12.70
12.10	12.10
11.90	11.90
11.80	11.80
12.80	12.80
13.40	13.40
10.60	10.60
11.20	11.20
12.10	12.10
9.70	9.70
18.00	18.00
18.00	18.00



X - Axis
Engine.Size.L.

Y - Axis
Fuel.Consumption.City..L.100.km.

☑ Show Regression Line
☐ Show Factors by Colours

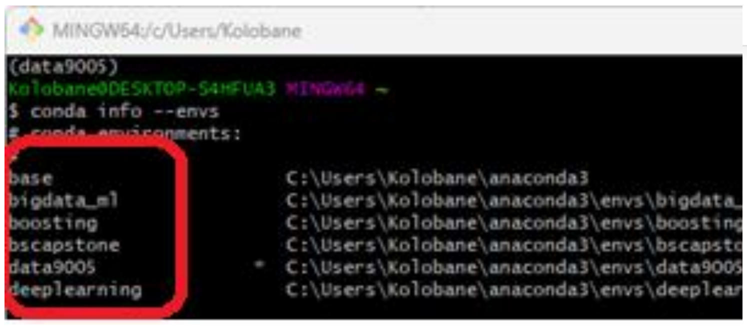
The coeffient is 5.766 and the slope is 2.149

Plot of Predicators
Histogram
Summary
Plot Generation
Scatter Plot of Fuel.Consumption.City..L.100.km. vs Engine.Size.L.
Fuel.Consumption.City..L.100.km.
Engine.Size.L.

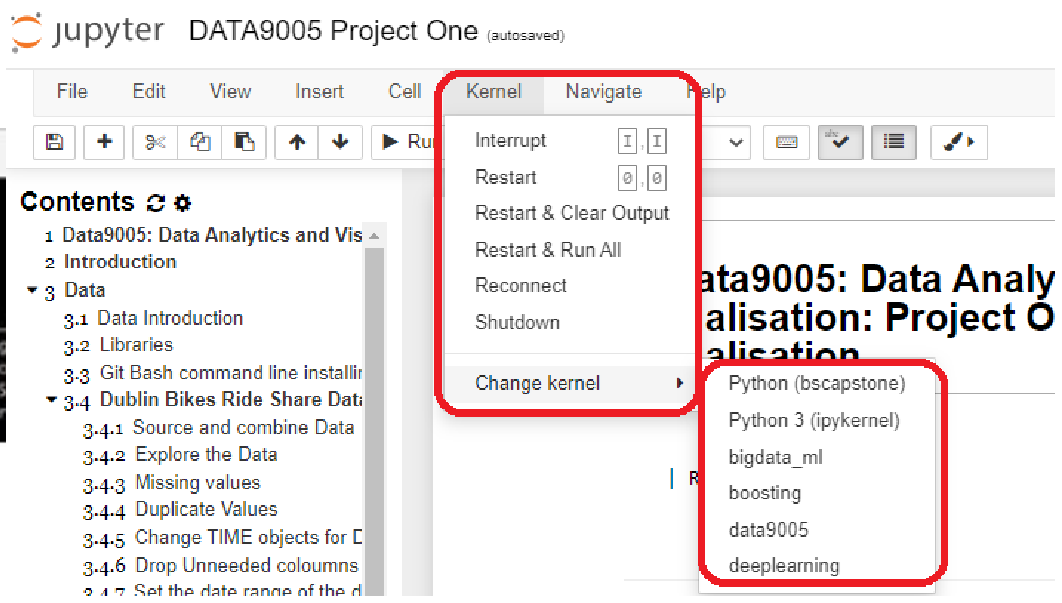
10. Additional work

10.1 Gitbash Kernel creation – Novel Concepts

I used Git Bash terminal commands to create my own Kernels in Jupyter Notebook.



```
MINGW64:/c/Users/Kolobane
(data9005)
Kolobane@DESKTOP-S4HFUA3 MINGW64 ~
$ conda info --envs
# conda environments:
#
base                     C:\Users\Kolobane\anaconda3
bigdata_ml               C:\Users\Kolobane\anaconda3\envs\bigdata_
boosting                 C:\Users\Kolobane\anaconda3\envs\boosting
bscapstone               C:\Users\Kolobane\anaconda3\envs\bscapsto
data9005              *  C:\Users\Kolobane\anaconda3\envs\data9005
deeplearning             C:\Users\Kolobane\anaconda3\envs\deeplear
```

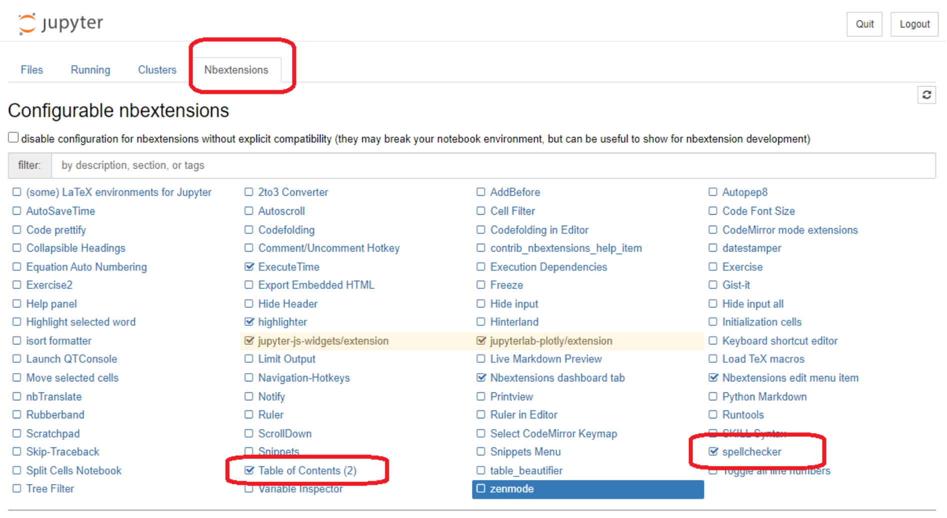


Contents
1 Data9005: Data Analytics and Vis
2 Introduction
3 Data
3.1 Data Introduction
3.2 Libraries
3.3 Git Bash command line installi
3.4 Dublin Bikes Ride Share Dat
3.4.1 Source and combine Data
3.4.2 Explore the Data
3.4.3 Missing values
3.4.4 Duplicate Values
3.4.5 Change TIME objects for D
3.4.6 Drop Unneeded coloumns

ata9005: Data Analy
alisation: Project O

10.2 Jupyter Notebook Nbextensions – Novel Concept

A table of Contents is essential for a large Jupyter Project. There is a great package that allows you to install additional extensions that will give you a Table of Contents, execution speeds, Spellchecker, etc.

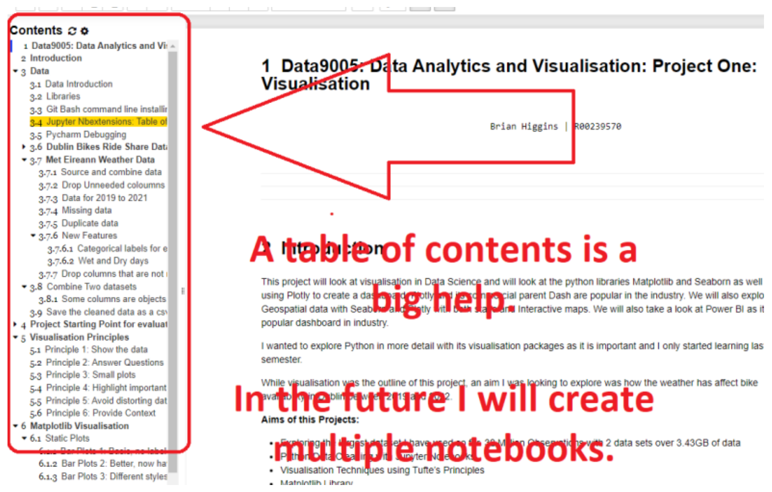


Configurable nbextensions

☐ disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)

filter: by description, section, or tags

☐ (some) LaTeX environments for Jupyter	☐ 2to3 Converter	☐ AddBefore	☐ Autopep8
☐ AutoSaveTime	☐ Autoscroll	☐ Cell Filter	☐ Code Font Size
☐ Code prettify	☐ Codefolding	☐ Codefolding in Editor	☐ CodeMirror mode extensions
☐ Collapsible Headings	☐ Comment/Uncomment Hotkey	☐ contrib_nbextensions_help_item	☐ datestamper
☐ Equation Auto Numbering	☑ ExecuteTime	☐ Execution Dependencies	☐ Exercise
☐ Exercise2	☐ Export Embedded HTML	☐ Freeze	☐ Gist-it
☐ Help panel	☐ Hide Header	☐ Hide input	☐ Hide input all
☐ Highlight selected word	☑ highlighter	☐ Hinterland	☐ Initialization cells
☐ isort formatter	☑ jupyter-js-widgets/extension	☑ jupyterlab-plotly/extension	☐ Keyboard shortcut editor
☐ Launch QTConsole	☐ Limit Output	☐ Live Markdown Preview	☐ Load TeX macros
☐ Move selected cells	☐ Navigation-Hotkeys	☑ Nbextensions dashboard tab	☑ Nbextensions edit menu item
☐ nbTranslate	☐ Notify	☐ Printview	☐ Python Markdown
☐ Rubberband	☐ Ruler	☐ Ruler in Editor	☐ Runtools
☐ Scratchpad	☐ ScrollDown	☐ Select CodeMirror Keymap	☐ SKILL Syntax
☐ Skip-Traceback	☐ Snippets	☐ Snippets Menu	☑ spellchecker
☐ Split Cells Notebook	☑ Table of Contents (2)	☐ table_beautifier	☐ Toggle all line numbers
☐ Tree Filter	☐ Variable Inspector	☐ zenmode	



Contents
1 Data9005: Data Analytics and Vi
2 Introduction
3 Data
3.1 Data Introduction
3.2 Libraries
3.3 Git Bash command line installi
3.4 Jupyter Nbextensions: Table of
3.5 Pycharm Debugging
3.6 Dublin Bikes Ride Share Dat
3.7 Met Eireann Weather Data
3.7.1 Source and combine data
3.7.2 Drop Unneeded coloumns
3.7.3 Data for 2019 to 2021
3.7.4 Missing data
3.7.5 Duplicate data
3.7.6 New Features
3.7.6.1 Categorical labels for e
3.7.6.2 Wet and Dry days
3.7.7 Drop columns that are not
3.8 Combine Two datasets
3.8.1 Some columns are objects
3.9 Save the cleaned data as a cs
4 Project Starting Point for evaluat
5 Visualisation Principles
5.1 Principle 1: Show the data
5.2 Principle 2: Answer Questions
5.3 Principle 3: Small plots
5.4 Principle 4: Highlight important
5.5 Principle 5: Avoid distorting dat
5.6 Principle 6: Provide Context
6 Matplotlib Visualisation
6.1 Static Plots
6.1.1 Bar Plots 1: Basic, no label
6.1.2 Bar Plots 2: Better, now ha
6.1.3 Bar Plots 3: Different styles

1 Data9005: Data Analytics and Visualisation: Project One: Visualisation

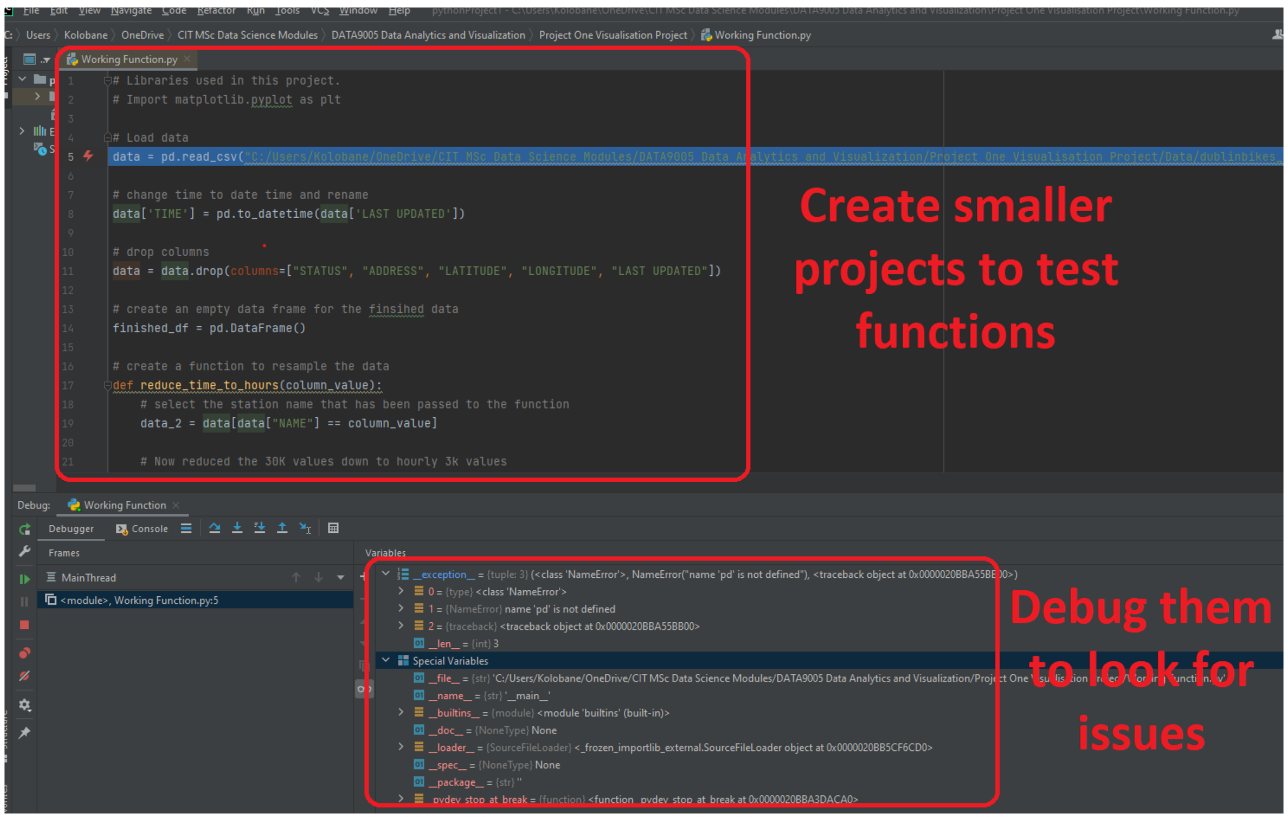
Brian Higgins | R00239570

A table of contents is a big help.

In the future I will create multiple notebooks.

11.3 Pycharm Coding and Debugging

Creating functions in Juypter is difficult as you cannot debug issues. I used Pycharm to create "mini projects" with data and then debug where the issues are and how the functions are working.



```
# Libraries used in this project.
# Import matplotlib.pyplot as plt

# Load data
data = pd.read_csv("C:/Users/Kolobane/OneDrive/CIT MSc Data Science Modules/DATA9005 Data Analytics and Visualization/Project One Visualisation Project/Data/dublinbikes_

# change time to date time and rename
data['TIME'] = pd.to_datetime(data['LAST UPDATED'])

# drop columns
data = data.drop(columns=["STATUS", "ADDRESS", "LATITUDE", "LONGITUDE", "LAST UPDATED"])

# create an empty data frame for the finsihed data
finished_df = pd.DataFrame()

# create a function to resample the data
def reduce_time_to_hours(column_value):
    # select the station name that has been passed to the function
    data_2 = data[data["NAME"] == column_value]

    # Now reduced the 30K values down to hourly 3k values
```

```
__exception__ = {tuple: 3} (<class 'NameError'>, NameError("name 'pd' is not defined"), <traceback object at 0x0000020BBA55BB00>)
    0 = {type} <class 'NameError'>
    1 = {NameError} name 'pd' is not defined
    2 = {traceback} <traceback object at 0x0000020BBA55BB00>
    __len__ = {int} 3
Special Variables
    __file__ = {str} 'C:/Users/Kolobane/OneDrive/CIT MSc Data Science Modules/DATA9005 Data Analytics and Visualization/Project One
    __name__ = {str} '__main__'
    __builtins__ = {module} <module 'builtins' (built-in)>
    __doc__ = {NoneType} None
    __loader__ = {SourceFileLoader} <_frozen_importlib_external.SourceFileLoader object at 0x0000020BB5CF6CD0>
    __spec__ = {NoneType} None
    __package__ = {str} ''
    pydev_stop_at_break = {function} <function pydev_stop_at_break at 0x0000020BBA3DACA0>
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help
Users › Kolobane › OneDrive › CIT MSc Data Science Modules › DATA9005 Data Analytics and Visualization › Project One Visualisation Project › Working Function.py
Working Function.py
Debug: Working Function
Debugger
Console
Frames
MainThread
<module>, Working Function.py:5
Variables
Create smaller projects to test functions
Debug them to look for issues

11.4 Functions – Novel Concept

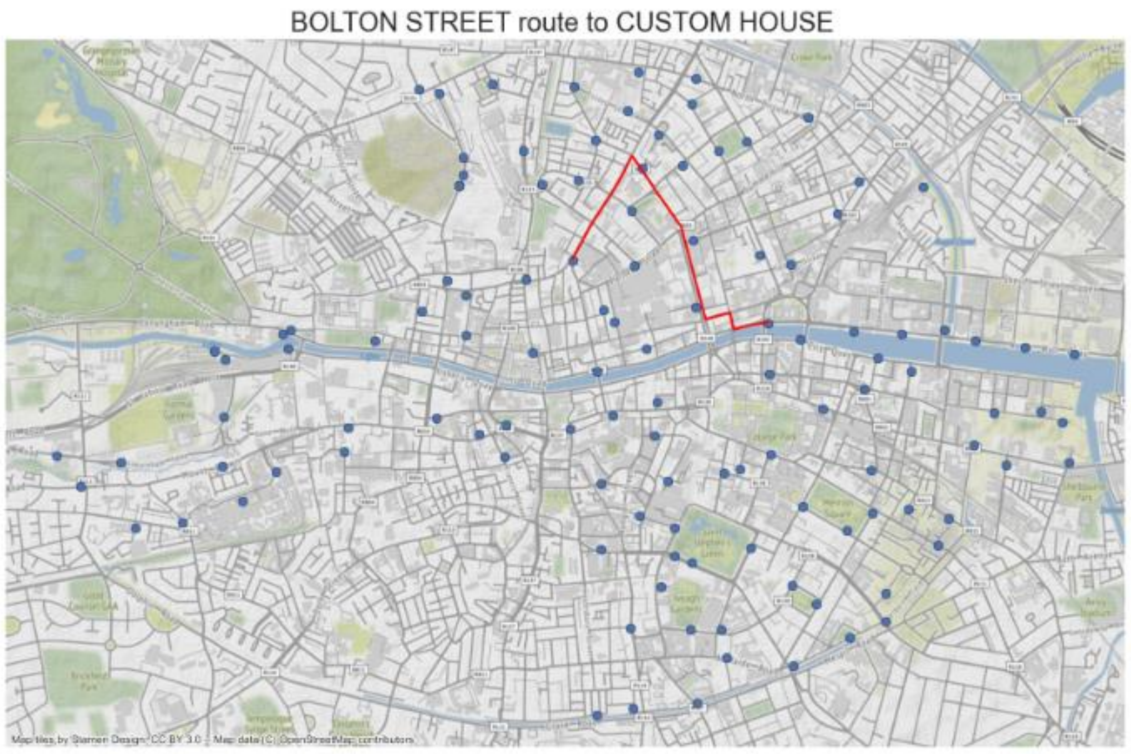
For this project, I created several functions, smaller ones to classify rain and temp data to create new features in notebook sections 4.2.6.1 and 4.2.6.2.

RESAMPLE FUNCTION. As already mentioned in section 4.1.8, this is used to reduce the 32 million observations taken several times an hour and resample down to the mean average of values per hour. This reduces the number of rows to 3 million.

A **ROUTE FUNCTION** that lets a user input any of the station's names and it will output a route between these stations using the OpenStreetMap API to send the Geometry points and return a route. (8.2.12)

NOTE: This command is **commented out** because the code will stop at this point. Seen working in the PDF or uncomment to run.

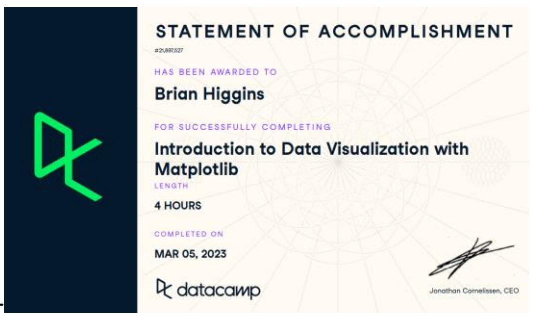
```
Map out the route between stations


Here is a list of Bike Stations:
        High Street, Custom House, Smithfield, Greek Street
        Dame Street, Hatch Street, Bolton Street


Enter start station: Bolton Street
Enter end station: Custom House
```


BOLTON STREET route to CUSTOM HOUSE
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

11. Data Camp Courses

Data Camp courses to get to know Python where I have completed 15 courses so far including the below 3 visualisation courses and am currently working on others:

- Understanding Data Visualisation
- Introduction to Data Visualisation with Matplotlib
- Introduction to Data Visualisation with Seaborn
- Intermediate Data Visualisation with Seaborn (Ongoing)



STATEMENT OF ACCOMPLISHMENT

#21,897,527

HAS BEEN AWARDED TO

Brian Higgins

FOR SUCCESSFULLY COMPLETING

Introduction to Data Visualization with Matplotlib

LENGTH

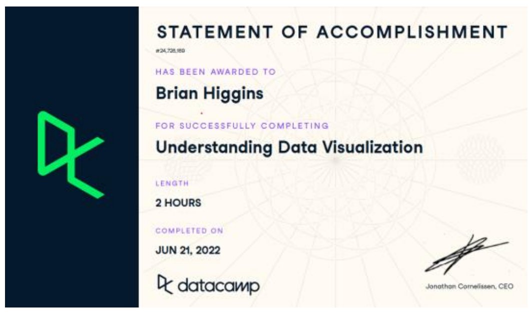
4 HOURS

COMPLETED ON

MAR 05, 2023

datacamp

Jonathan Cornelissen, CEO



STATEMENT OF ACCOMPLISHMENT

#24,728,169

HAS BEEN AWARDED TO

Brian Higgins

FOR SUCCESSFULLY COMPLETING

Understanding Data Visualization

LENGTH

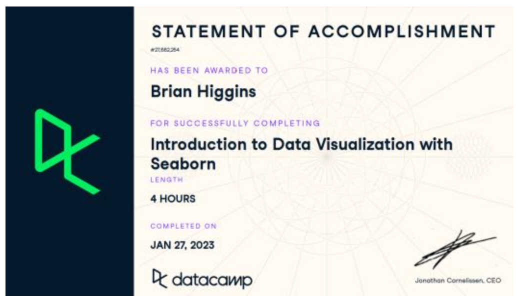
2 HOURS

COMPLETED ON

JUN 21, 2022

datacamp

Jonathan Cornelissen, CEO



STATEMENT OF ACCOMPLISHMENT

#27,582,254

HAS BEEN AWARDED TO

Brian Higgins

FOR SUCCESSFULLY COMPLETING

Introduction to Data Visualization with Seaborn

LENGTH

4 HOURS

COMPLETED ON

JAN 27, 2023

datacamp

Jonathan Cornelissen, CEO

12. Future Work and What I learned

- **Jupyter Notebook:** Mine is too big, next time I'll break it up into several notebooks.
- **Data:** I wanted to work on large data, time series and geospatial which I need. But in future, I need to be aware if the data has other features I can use, as some of the Dublin Data was not public.
- I choose **Plotly** to make a dashboard but this was much harder to host online than I thought, this will take more studying.
- Future areas would include Analysis with Time Series Algorithms and Clustering of stations.[1]

13. Other Projects

There are projects on GitHub where students have created apps to show the availability of bikes at stations. I didn't find these that useful, as a lot were more interested in the app side than the visualisation side. Also, there were not many (bare one I found) that had the extra features I was missing but I could not use their data.

Some Analysis was also done by looking at clustering [1] as well as bike usage patterns with clustering [2]. A large thesis showed some greater insight into how Dublin Bikes can be integrated into a Smart City [3].

14. References

[1] R. Breslin, "What Dublin Bikes data can tell us about the city and its people," *Medium*, Dec. 20, 2020. https://towardsdatascience.com/what-dublin-bikes-data-can-tell-us-about-the-city-and-its-people-63fde77ee383 (accessed Mar. 06, 2023).

[2] James, "Usage patterns of Dublin Bikes stations," *Medium*, Dec. 19, 2017. https://towardsdatascience.com/usage-patterns-of-dublin-bikes-stations-484bdd9c5b9e (accessed Mar. 06, 2023).

[3] H. Ul Ahad, "Dublin Smart City Data Integration, Analysis and Visualisation," Technological University Dublin, 2020. doi: 10.21427/5EKZ-6D18.

15. Appendix A

14.1 Technical Item List

Technical Item checklist of different tasks completed in this project with their location. The note of “complex” is used where I found things took a larger amount of time (hours) or I found added extra complexity with APIs, functions, etc.

Jupyter Notebook	• Kernel Creation (3.3) • Nbextensions(3.4) -Table of contents -Spellcheck -Runtime • Pycharm Debugging for Complex Functions(3.5)	**Matplotlib**	• Bar Plots (3) (7.1.1-7.1.3) • Line Plots (2) (7.1.4/7.1.5) • Scatter Plots (7.1.6) • Histograms (7.1.7) • Box Plots (2) (7.1.8/7.1.9) • Multiple Plots (7.1.10) • Pie Charts (7.1.11) • 3D Plots (7.1.12)
Data Processing	• Combine Datasets (4.1.1) • Missing values (4.1.3) • Duplicates (4.1.3) • Change object types (4.1.5) • Drop columns (4.1.6) • Functions: - Reduce data from 32million to 3million (4.1.8) - Create new features for rain and temperature (4.2.6.1/4.2.6.2)	**Seaborn**	• Bar Plots (8.1.1) • Count Plots (8.1.2) • Line Plots (5) (8.1.3-8.1.7) • Scatter Plots (2) (8.1.8/8.1.9) • Histogram (3) (8.1.11-8.1.13) • Boxplot (3) (8.1.14-8.1.16) • Heat map (8.1.17)
Feature Creation	• Time of the day (4.1.9.1) • Hour of the day (4.1.9.2) • Days of the week on numeric and categorical types (4.1.9.4 /4.1.9.5) • Month feature (4.1.9.6) • Year feature (4.1.9.7) • Percentage of bikes available (4.1.9.9)	**Geospatial Maps**	• Scatter Plot (8.2.4) • 2 Layer Plot (8.2.5) • Show different base maps (8.2.6-8.28) • Routes (8.2.9 -8.2.11) • Route Function to map routes for user (8.2.12) • Polygon mapping (8.2.13) • Route Function (8.2.12)
Combine datasets	• Combine Bike and Met Eireann data (4.3)	**Dashboard**	• Plotly Static Plots (9.2.2.1-9.2.2.4) • Dashboards (3) - Plots (9.4.1) - Tabs (9.4.2) - Maps (9.4.3) - Themes (9.4.3

14.2 Jupyter Notebook Table of Contents.