**Code documentation**

Below is a guide to what each bit of R code/script/function does

**Read.googlemobility.R:**

1. Reads in the most recent Google mobility report from<https://www.google.com/covid19/mobility/>
2. Creates a subset of data for the United Kingdom.
3. Manually creates a new column defining the country (England, Wales, Scotland, and Northern Ireland) for each sub region as defined using:
   1. <https://en.wikipedia.org/wiki/Subdivisions_of_England>
   2. <https://en.wikipedia.org/wiki/Local_government_in_Wales>
   3. h[ttps://en.wikipedia.org/wiki/Subdivisions\_of\_Scotland](https://en.wikipedia.org/wiki/Subdivisions_of_Scotland)
   4. <https://en.wikipedia.org/wiki/Local_government_in_Northern_Ireland>

**google\_england.csv:** Read.***googlemobility.R*** is ran to obtain the Google mobility report and then subsetted for England districts and saved as a csv.

**Read.metofficecovid.R:**

1. Reads in all the files located under the <https://metdatasa.blob.core.windows.net/covid19-response/> file branch.
2. Creates a merged dataset of all the available datasets under this branch to produce a continuous time series.

**Match.metoffice.R:**

1. Both the ***google\_england.csv*** and the output from the ***read.metofficecovid.R*** is read into this function as google\_df and metoffice\_df respectively.
2. A new column is manually created for the metoffice\_df defining the country (England, Wales, Scotland, and Northern Ireland) for each sub region using:
   1. <https://en.wikipedia.org/wiki/Subdivisions_of_England>
   2. <https://en.wikipedia.org/wiki/Local_government_in_Wales>
   3. h[ttps://en.wikipedia.org/wiki/Subdivisions\_of\_Scotland](https://en.wikipedia.org/wiki/Subdivisions_of_Scotland)
   4. <https://en.wikipedia.org/wiki/Local_government_in_Northern_Ireland>
3. A new data frame known as metoffice\_england is created by subsetting out all England defined sub regions.
4. A new column is added to metoffice\_england to define the google named regions that match or are located in the sub regions defined by metoffice\_england.
5. If there were many metoffice regions that belonged to one google defined region these were aggregated by mean.
6. Metoffice variables for “precipitation flux” and “air temperature” were converted to mm/hr and Celsius respectively for comparability with OpenWeather data used for forecasting.

**Metoffice\_england.csv:** Output from match.metoffice.R written as a csv.

**Relative2baseline.R:**

1. Reads in the Metoffice\_england.csv or the output dataset from ***Getandmatch.forecast.R*** (discussed later.)
2. Extracts all data points that exist after the 15/02/2020 (beginning of the Google mobility datasets.)
3. Reads in a dataset that contains the baseline values for the Metoffice data corresponding to the baseline calculation for the Google Mobility report. (Median value per weekday from between 03/01/2020 and 06/02/2020)
4. Subsets out the metoffice data frame to contain only temperature (mean, max, and minimum) and rain (mean) variables that are compatible with the OpenWeather data.
5. Creates a data frame for the percentage change relative to their baseline for each meteorological variable by considering the meteorological variable, the sub region, and the weekday.

**Googleandmetoffice\_england.csv:** Merge of the ***google\_england.csv*** and the output from ***relative2baseline.R*** for the ***Metoffice\_england.csv***, written as a csv.

**Create.model.R:**

1. Reads in the ***googleandmetoffice\_england.csv***, the ***mene\_england.csv***, and the ***gardenaccess\_england.csv****.*
2. Subsets out all the above datasets to only contain variables required for the model which were predefined by exploratory analysis.
3. Combines all the subsetted data sets together.
4. A model is created by running this dataset through randomForest.

**RF\_model.RDS:** Model outputted from the ***create.model.R***function and saved as an RDS.

**Getandmatch.forecast.R:**

1. Requires a defined location, as specified by the Google subregions and an API key from OpenWeather.
2. All locations’ forecasts are extracted from the Open Weather server using the metoffice defined sub regions names.
3. Data points are collected for these metoffice defined sub regions and then converted into the Google defined sub regions.
4. Aggregation of the mean may occur when many metoffice defined sub regions equate to one Google defined sub region.
5. The Open weather dataset is retrieved and then subsetted for the required meteorological variables (temperature and rainfall), and rainfall is converted into mm/hr.
6. Each meteorological variable is aggregated across each day by:
   1. Mean for mean temperature and mean rain.
   2. Maximum for max temperature.
   3. Minimum for minimum temperature.
7. The***mene\_england.csv***, and the ***garden\_access.csv*** is read in, where variables required by the model are inserted into the data frame for their defined location.

**Forecast\_england.csv:** A combined dataframe for the output for each Google defined sub region as the location in the ***Getandmatch.forecast.R*** function, then ran through ***relative2baseline.R*** function written as a csv.

**Plot.googlemobilitydistricts.R:**

1. This code has argumentsspecifying:
   1. Dataset by ‘google’ (***googleandmetoffice\_england.csv*)**
   2. Type of place by ‘type’ (default = “parks”)
   3. District to be plotted for by ‘district’ (default = “Bedford”)
   4. Plotting rain trend by ‘rain’ (default = T)
   5. Plotting temperature trend by ‘temp’ (default = T)
2. **Scaling factors** for the trend lines are calculated for both mean temperature and mean rainfall by considering both positive and negative percentage changes:
   1. **For positive:** Identifying the maximum percentage change for mobility and dividing by the maximum percentage change for temperature or rainfall.
   2. **For Negative:** Identifying the minimum percentage change for mobility and dividing by the minimum percentage change for temperature or rainfall.
3. A column is added to the dataset, known as “colour\_choice” specifying whether the parks visitation percentage change is positive or negative and denoted by colours “darkgreen” and “grey” respectively.
4. A subset of the dataset is created by extracting data points for the specified ‘district’ and selecting columns required for plotting.
5. The scaled temperature and rainfall are calculated by adding a column specifying the scaling factor required for if the variables percentage change is negative or positive.
6. Values in “temp\_mean” and “rain\_mean” are converted by multiplying them by their predefined scaling factor.
7. If the arguments of ‘rain’ or ‘temp’ is FALSE, their values will be replaced with 0’s to ensure a trend-line is not plotted.
8. A graph is created using ggplot where park visitation percentage change is plotted as a bar graph with
   1. The y-axis being replaced for i.e. 2x instead of +100%.
   2. Percentage change increases are dark green bars, decreases are grey.
   3. Trend lines for rain and temperature are plotted using geom\_line.

**Plot.parkvisits.R:**

1. This code has argumentsspecifying:
   1. Dataset by ‘googleandmetoffice’ (***googleandmetoffice\_england.csv*)**
   2. Model by ‘model’ (**RF\_model.RDS**)
   3. Forecasting dataset by ‘forecast’ (**Forecast\_england.csv)**
   4. District by district (default = “Bedford”)
   5. dayofweek = wday (default = wday(as.Date(Sys.Date())) )
   6. Plot.parkvisits (default = F)
   7. Plotting rain trend by ‘rain’ (default = T)
   8. Plotting temperature trend by ‘temp’ (default = T)
2. Nextweekday function is used from Kirill (<https://stackoverflow.com/questions/32434549/how-to-find-next-particular-day>)
3. The weekday as specified by the dayofweek is converted into it’s character version.
4. The ***googleandmetoffice\_england.csv*** is subsetted for relevant factors and the scaling factors are computed as specified in **Plot.googlemobilitydistricts.R.**
5. Missing values for the parks visitation percent change for the specified weekday in the ***googleandmetoffice\_england.csv*** are identified and additionally those that don’t have weather data are removed.
6. **Forecast\_england.csv** is read in and cleaned to only contain district specific data points and variables required for prediction through the model.
7. A forecast\_district vector is made by containing modelling variables that are consistent per district and is merged with the missing data identified to allow for prediction through the model.
8. The missing data and forecasting data is run through the model, separately, to obtain a dataframe with prediction values for park visitation percentage change, date, temp\_mean and rain\_mean values. A vector of NA’s is curated in place if:
   1. There is no missing data in the historical mobility data.
   2. There is no data for the selected weekday in the forecasting dataset.
9. If the missing dataset contains a NA for date OR predict\_missing = FALSE, then a new dataframe is created with only the forecasting predicting data, if not both the forecasting prediction dataset and the missing data prediction dataset is combined. This dataset is known as **prediction\_row**.
10. A new dataframe, called **google\_metoffice\_current\_district\_and\_weekday**, is created from the ***googleandmetoffice\_england.csv*** to contain all values for park visitation percentage change for the specified day and district.
11. A column is added to both **prediction\_row** and **google\_metoffice\_current\_district\_and\_weekday** known as colour\_choice to specify the colours of the bars.
    1. Prediction values are assigned “white”
    2. Positive actual values are assigned “darkgreen”
    3. Negative actual values are assigned “grey”
12. **google\_metoffice\_current\_district\_and\_weekday** is created to contain no NA values, except for those obtained after the date for which metoffice data is not available.
13. Both **prediction\_row** and **google\_metoffice\_current\_district\_and\_weekday** are combined to create a full dataset for plotting.
14. Scaling values are added to the dataset as outlined in **Plot.googlemobilitydistricts.R.**
15. An aes dataframe is curated from the colours present in the colour\_choice column and each are assigned their complimentary label for the plot.
16. Lastly the graph is curated as outlined by **Plot.googlemobilitydistricts.R.** plus additionally using the aes dataframe to create a figure legend corresponding to each coloured bar graph.