GY7702 Assignment 1

Ben Coombs

14/11/2020

R for Data Science - GY7702 Assignment 1

This document is my submission for Assignment 1 for R for Data Science (GY7702). It was created using RMarkdown in RStudio and knitted into a pdf. You can find more information in my **Github repository**.

Loading libraries

Throughout this assignment, I will be making use of the libraries `tidyverse` and `knitr`. Therefore it is a good idea to load them straight away.

```
library(tidyverse)
library(knitr)
```

Question 1

Q1.1: write the code necessary to check whether all participants to the survey either completely disagree or completely agree, once the missing values are excluded.

```
#Create the vector of 25 numbers listed on the question paper
nums <- c(NA, 3, 4, 4, 5, 2, 4, NA, 6, 3, 5, 4, 0, 5, 7, 5, NA, 5, 2, 4, NA,
          3, 3, 5, NA)
#Create a new vector of the same numbers, this time with missing values (NA
#values) omitted
nums_new <- nums[!is.na(nums)]
#Check if all responses are strongly agree or strongly disagree
all(nums_new %>% is.element(c(1,7)))
```

```
## [1] FALSE
```

A return of `FALSE` indicates that there were participants in the survey that did not either completely agree or completely disagree.

Q1.2: write the code necessary to extract the indexes related to the participants in the survey who at least somehow agree or more.

```r
#Return the positions in the vector of elements that are participants responding
#somehow agree or stronger (5 or greater).
which(nums_new >= 5)
```

```
## [1]  4  7  9 12 13 14 15 20
```

Question 2

Q2.1: Install the library `palmerpenguins`

```r
#Load the installed library "palmerpenguins"
library(palmerpenguins)
```

Q2.2: write the code necessary to create a table showing *species*, *island*, *bill length* and *body mass* of the 10 *Gentoo* penguins in the penguins table with the highest *body mass*.

```r
#Create a table to show species, island, bill length and body mass of the 10
#Gentoo penguins with the highest body mass
gentoo_penguins <- penguins %>%
  #Only want the table to show species, island, bill length and body mass
  select(species, island, bill_length_mm, body_mass_g) %>%
  #Only display rows of Gentoo penguins
  filter(species == "Gentoo") %>%
  #Arrange the rows by body mass largest to smallest
  arrange(-body_mass_g) %>%
  #Take the first 10 values, i.e. the 10 Gentoo penguins with the largest body
  #mass
  slice_head(n = 10)

#Display the table
kable(gentoo_penguins)
```

species	island	bill_length_mm	body_mass_g
Gentoo	Biscoe	49.2	6300
Gentoo	Biscoe	59.6	6050
Gentoo	Biscoe	51.1	6000
Gentoo	Biscoe	48.8	6000
Gentoo	Biscoe	45.2	5950
Gentoo	Biscoe	49.8	5950
Gentoo	Biscoe	48.4	5850
Gentoo	Biscoe	49.3	5850
Gentoo	Biscoe	55.1	5850
Gentoo	Biscoe	49.5	5800

Q2.3: write the code necessary to create a table showing the average *bill length* per *island*, ordered by average *bill length*.

```
#Create a table of the average bill length per island ordered by average bill
#length
island_bill_length <- penguins %>%
  #Only considering island and bill length
  select(island, bill_length_mm) %>%
  #Remove NA values to correctly summarise
  filter(!is.na(bill_length_mm)) %>%
  #Group rows by island
  group_by(island) %>%
  #Summarise the islands by the average bill length
  summarise(
    avg_bill_length_mm = mean(bill_length_mm)
  ) %>%
  #Arrange the table by highest to lowest average bill length
  arrange(-avg_bill_length_mm)

#Display the table
knitr::kable(island_bill_length)
```

island	avg_bill_length_mm
Biscoe	45.25749
Dream	44.16774
Torgersen	38.95098

Q2.4: write the code necessary to create a table showing the minimum, median and maximum proportion between *bill length* and *bill depth* by *species*.

```
#Create a table to show the maximum, median and minimum proportion between bill
#length and bill depth by species
bill_ratios <- penguins %>%
  #Only considering species, bill length and bill depth
  select(species, bill_length_mm, bill_depth_mm) %>%
  #Remove any rows with NA values to correctly summarise
  filter(!is.na(bill_length_mm), !is.na(bill_depth_mm)) %>%
  #Calculate and create a column for the bill ratio of each penguin
  mutate(
    bill_ratio = bill_length_mm / bill_depth_mm
  ) %>%
  #Group rows by species
  group_by(species) %>%
  #Summarise each species with their minimum, median, and maximum bill ratios
  summarise(
    min_proportion = min(bill_ratio),
    median_proportion = median(bill_ratio),
    max_proportion = max(bill_ratio)
  )
```

```
#Display the table
knitr::kable(bill_ratios)
```

species	min_proportion	median_proportion	max_proportion
Adelie	1.639810	2.136842	2.450000
Chinstrap	2.350516	2.661577	3.258427
Gentoo	2.566474	3.166667	3.612676

Question 3

Q3.1: write the code necessary to load the data from `covid19_cases_20200301_20201017.csv` to a variable named `covid_data`.

`covid19_cases_20200301_20201017.csv` contains information **from the UK Government** on the number of new and cumulative cases of covid-19 from 1st March to 17th October in the UK.

```
#Load the csv data to a variable
covid_data <- read_csv("covid19_cases_20200301_20201017.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   specimen_date = col_date(format = ""),
##   area_name = col_character(),
##   newCasesBySpecimenDate = col_double(),
##   cumCasesBySpecimenDate = col_double()
## )
```

This message indicates that the CSV has been successfully loaded.

Q3.2: write the code necessary to:

- Create a complete table, containing a row for each day and area.

- Replace `NA` values with the value available for the previous date.

- Replace the remaining `NA` values (those that don't have a previous value available) with zero.

- Subset only the area assigned to your student ID in the table in the appendix.

- Drop the `area_name` column.

```r
#Store this complete table that is being created into the variable
#named Ealing_complete_covid_data
Ealing_complete_covid_data <- covid_data %>%
  #Create complete table with a row for each day and area
  tidyr::complete(specimen_date, area_name) %>%
  #Arrange the table by area name (A-Z) and specimen date (oldest to newest)
  arrange(area_name, specimen_date) %>%
  #Group by area name to avoid the risk of filling NA values in a later step
  #with values from a different area
  group_by(area_name) %>%
  #Fill NA values of new and cumulative cases with the value of the previous
  #date for that area, if possible
  tidyr::fill(newCasesBySpecimenDate, cumCasesBySpecimenDate) %>%
  #Replace any remaining NA values with 0
  tidyr::replace_na(list(newCasesBySpecimenDate = 0,
                         cumCasesBySpecimenDate = 0)
                    ) %>%
  #Subset the assigned area
  filter(area_name == "Ealing") %>%
  #Ungroup to drop area_name column
  ungroup() %>%
  #Drop area_name column
  select(-area_name)

#Display the first five rows
Ealing_complete_covid_data %>%
  slice_head(n = 5) %>%
  kable()
```

- Store the resulting table in a variable named `[area]_complete_covid_data` (substituting `[area]` in the name of the variable with the name of the area assigned to you).

specimen_date	newCasesBySpecimenDate	cumCasesBySpecimenDate
2020-03-01	0	1
2020-03-02	1	2
2020-03-03	1	3
2020-03-04	1	4
2020-03-05	1	5

Q3.3: Starting from the table `[area]_complete_covid_data` created for *Question 3.2*:

- Create a copy of `[area]_complete_covid_data`, i.e., as another variable named `[area]_day_before`.

- Use the library `lubridate` to create a new column named `day_to_match` in the new table `[area]_day_before` that reports the day after the day reported in the column `specimen_date`, as a character value (e.g., if `specimen_date` is `"2020-10-10"`, `day_to_match` should be `"2020-10-11"`). This way, the cases registered on Oct 10th will be used as point of comparison for the cases registered on Oct 11th.

- Drop the `specimen_date` and `cumCasesBySpecimenDate` columns from the `[area]_day_before` table.

- Rename the `newCasesBySpecimenDate` column of the the `[area]_day_before` table to `newCases_day_before`.

- Join `[area]_day_before` with `[area]_complete_covid_data`, where the column `specimen_date` of `[area]_complete_covid_data` is equal to the column `day_to_match` of `[area]_day_before`.

- Calculate a new column in the joined table, containing the number of new cases as a percentage of the number of new cases of the day before.

- Store the resulting table in a variable named `[area]_covid_development`.

```r
#Create a copy of Ealing_covid_data as a new variable
Ealing_day_before <- Ealing_complete_covid_data %>%
  #Create new column reporting the day after the date reported in Specimen_date
  #column
  mutate(
    day_to_match = specimen_date + 1
  ) %>%
  #Drop the columns specimen_date and cumCasesBySpecimenDate
  select(-specimen_date, -cumCasesBySpecimenDate) %>%
  #Rename the newCasesBySpecimen date to newCases_day_before
  rename(newCases_day_before = newCasesBySpecimenDate)

#Store this table that is being created into the variable named
#Ealing_covid_development
Ealing_covid_development <- Ealing_complete_covid_data %>%
  #Join this table with Ealing_complete_covid_data where day_to_match is equal
  #to specimen_date
  full_join(
    Ealing_day_before,
    by = c("specimen_date" = "day_to_match")
  ) %>%
  #Create new calculated column of new cases as a percentage of the number of
  #cases of the previous day
  mutate(
    As_percentage_of_day_before = (
      newCasesBySpecimenDate/newCases_day_before
      )*100
  )
```

```r
#Display the first five lines of each table
Ealing_day_before %>%
  slice_head(n = 5)
```

```
## # A tibble: 5 x 2
##   newCases_day_before day_to_match
##                 <dbl> <date>
## 1                   0 2020-03-02
## 2                   1 2020-03-03
## 3                   1 2020-03-04
## 4                   1 2020-03-05
## 5                   1 2020-03-06
```

```r
Ealing_covid_development %>%
  slice_head(n = 5)
```

```
## # A tibble: 5 x 5
##   specimen_date newCasesBySpeci~ cumCasesBySpeci~ newCases_day_be~
##   <date>                   <dbl>            <dbl>            <dbl>
## 1 2020-03-01                   0                1               NA
## 2 2020-03-02                   1                2                0
## 3 2020-03-03                   1                3                1
## 4 2020-03-04                   1                4                1
## 5 2020-03-05                   1                5                1
## # ... with 1 more variable: As_percentage_of_day_before <dbl>
```

Days reporting 0 new cases lead to `Inf` being the value for `As_percentage_of_day_before` of the following day as R is being asked to divide by zero. Instead, I will replace these values with `NaN`.

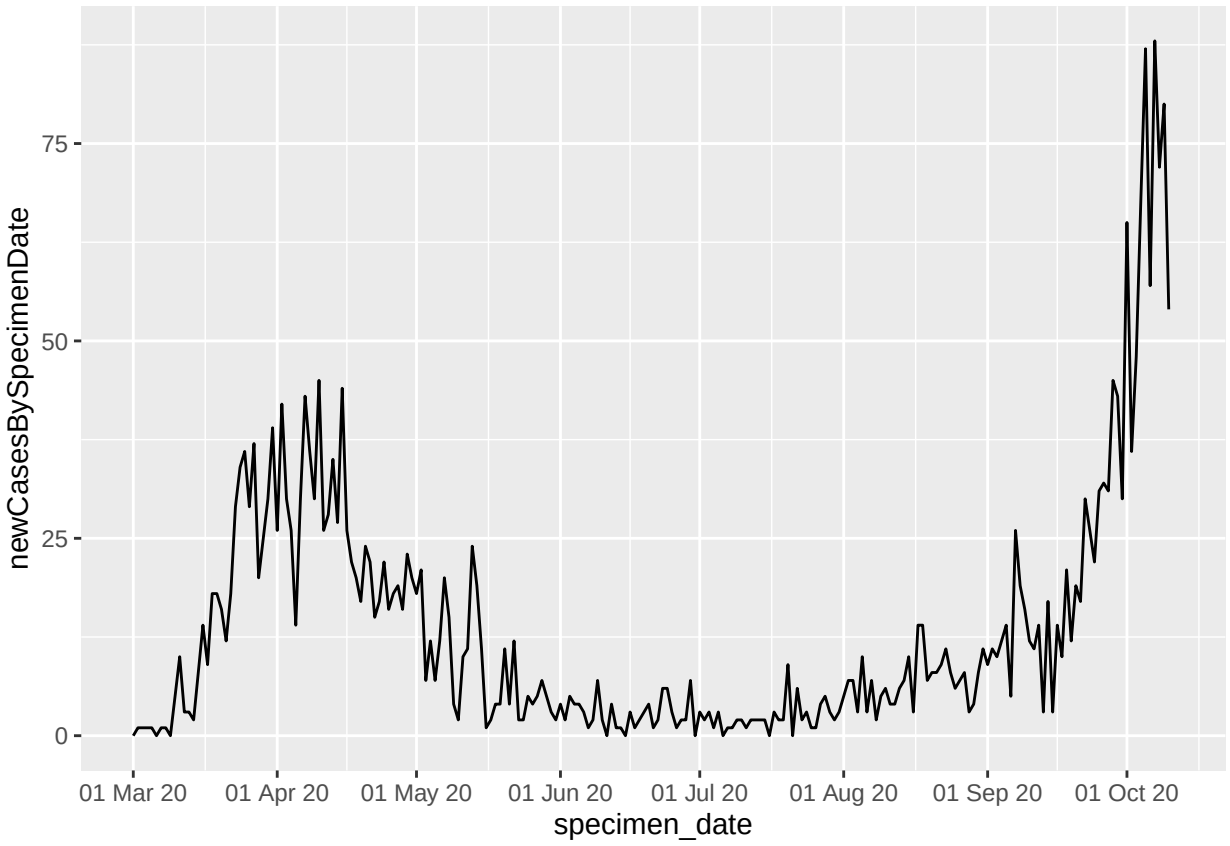
```r
#Replace "Inf" values in As_percentage_of_day_before with NaN
Ealing_covid_development$As_percentage_of_day_before[
  is.infinite(Ealing_covid_development$As_percentage_of_day_before)
  ] <- NaN

#Display the first 10 lines
Ealing_covid_development %>%
slice_head(n = 5)
```

```
## # A tibble: 5 x 5
##   specimen_date newCasesBySpeci~ cumCasesBySpeci~ newCases_day_be~
##   <date>                   <dbl>            <dbl>            <dbl>
## 1 2020-03-01                   0                1               NA
## 2 2020-03-02                   1                2                0
## 3 2020-03-03                   1                3                1
## 4 2020-03-04                   1                4                1
## 5 2020-03-05                   1                5                1
## # ... with 1 more variable: As_percentage_of_day_before <dbl>
```

Q3.4: Write a short text (max 150 words) describing the development of new cases in the area over time, as evidenced by the table [area]_covid_development.

```
ggplot(Ealing_covid_development,
       aes(x = specimen_date, y = newCasesBySpecimenDate)
       ) +
  geom_line() +
  scale_x_date(date_breaks = "1 month", date_labels = "%d %b %y")
```



From the plot above, we can see that cases started to increase rapidly from zero from early to mid-March, reaching on average around 30 new cases per day. When the initial lockdown was introduced, the rate of increase of new cases per day started to decline, though the variation in new cases per day data is quite large. The number of new daily cases then began to follow a downward trend from mid-April, with the rate of decrease slowly tapering off until daily new cases sustained an average of just under 5 between the beginning of June and start of August. From August, daily new cases began to rise exponentially, passing April levels by late September and reaching almost 90 new cases in one day in early October.

Question 4

```
#Load the csv data to a variable
lad_pop <- read_csv("lad19_population.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   lad19_area_name = col_character(),
```

```
##   lad19_area_code = col_character(),
##   area_population = col_double()
## )
```

This message indicates that the CSV has been successfully loaded.

```
#Join the local authority info with the covid_data
lad_covid <- lad_pop %>%
  full_join(
    covid_data,
    by = c("lad19_area_name" = "area_name")
  )
```

I want use the population data for Ealing from `lad_covid` to calculate a new column `Ealing_covid_development` that reports the number of new cases per 100,000 of the population for that day. This will allow for comparison with other areas.

```
Ealing_covid_pop <- lad_covid %>%
  #Include only rows related to Ealing from lad_covid
  filter(lad19_area_name == "Ealing") %>%
  #Consider only the area_population and specimen_date column
  select(area_population, specimen_date) %>%
  #Join the area population details onto Ealing_covid_development from Q3.3
  full_join(
    Ealing_covid_development,
    by = c("specimen_date" = "specimen_date")
  ) %>%
  #Calculate new column reporting new daily cases per 100k
  mutate(
    new_cases_per_100k = (newCasesBySpecimenDate/area_population)*100000
  )
```

Obviously, in order to compare with another area, I need to obtain these figures for at least one other area. I will create a set of three functions that will be nested like a set of Russian dolls so that only a single function needs to be called to get the equivalent of `Ealing_covid_pop` for any area.

```
#First of all, I need the complete table containing a row for each day and area
complete_covid_data <- covid_data %>%
  #Create complete table with a row for each day and area
  tidyr::complete(specimen_date, area_name) %>%
  #Arrange the table by area name (A-Z) and specimen date (oldest to newest)
  arrange(area_name, specimen_date) %>%
  #Group by area name to avoid the risk of filling NA values in a later step
  #with values from a different area
  group_by(area_name) %>%
  #Fill NA values of new and cumulative cases with the value of the previous
  #date for that area, if possible
  tidyr::fill(newCasesBySpecimenDate, cumCasesBySpecimenDate) %>%
  #Replace any remaining NA values with 0
  tidyr::replace_na(list(newCasesBySpecimenDate = 0,
                         cumCasesBySpecimenDate = 0)
                    )
```

Now I can use this table to create a function to get complete covid data of any area, and save it to the variable `Area_complete_covid_data`.

```
Area_complete_covid_data <- function(area) {
  complete_covid_data %>%
  #Subset the assigned area
  filter(area_name == area) %>%
  #Ungroup to drop area_name column
  ungroup() %>%
  #Drop area_name column
  select(-area_name)
}
```

Now I can use this function to create another function to return the equivalent of `Ealing_covid_development` for any given area, and save it to the variable `Area_covid_development`.

```
Area_covid_development <- function(area) {
  covid_dev <- Area_complete_covid_data(area) %>%
  #Create new column reporting the day after the date reported in Specimen_date
  #column
  mutate(
    day_to_match = specimen_date + 1
  ) %>%
  #Drop the columns specimen_date and cumCasesBySpecimenDate
  select(-specimen_date, -cumCasesBySpecimenDate) %>%
  #Rename the newCasesBySpecimen date to newCases_day_before
  rename(newCases_day_before = newCasesBySpecimenDate) %>%
  #Join this table with Ealing_complete_covid_data where day_to_match is equal
  #to specimen_date
  full_join(
    Area_complete_covid_data(area),
    by = c("day_to_match" = "specimen_date")
  ) %>%
  #Create new calculated column of new cases as a percentage of the number of
  #cases of the previous day
  mutate(
    As_percentage_of_day_before = (
      newCasesBySpecimenDate/newCases_day_before
      )*100
  )

  #Replace "Inf" values in covid_dev with NaN
  covid_dev$As_percentage_of_day_before[
  is.infinite(covid_dev$As_percentage_of_day_before)
  ] <- NaN

  covid_dev
}
```

Now I can use Area_covid_development in a function to return the `Ealing_covid_pop` table for any area. I am going to use the function to get the daily new infection rate per 100,000 for the area Powys.

```
Area_covid_pop <- function(area) {
  covid_pop <- lad_covid %>%
    #Include only rows related to the input area
    filter(lad19_area_name == area) %>%
    #Take only area_population and specimen_date
    select(area_population, specimen_date) %>%
    #Join the area population details onto Ealing_covid_development from Q3.3
    full_join(
      Area_covid_development(area),
      by = c("specimen_date" = "day_to_match")
    ) %>%
    mutate(
      new_cases_per_100k = (newCasesBySpecimenDate/area_population)*100000
    )
}

Powys_covid_pop <- Area_covid_pop("Powys")
```

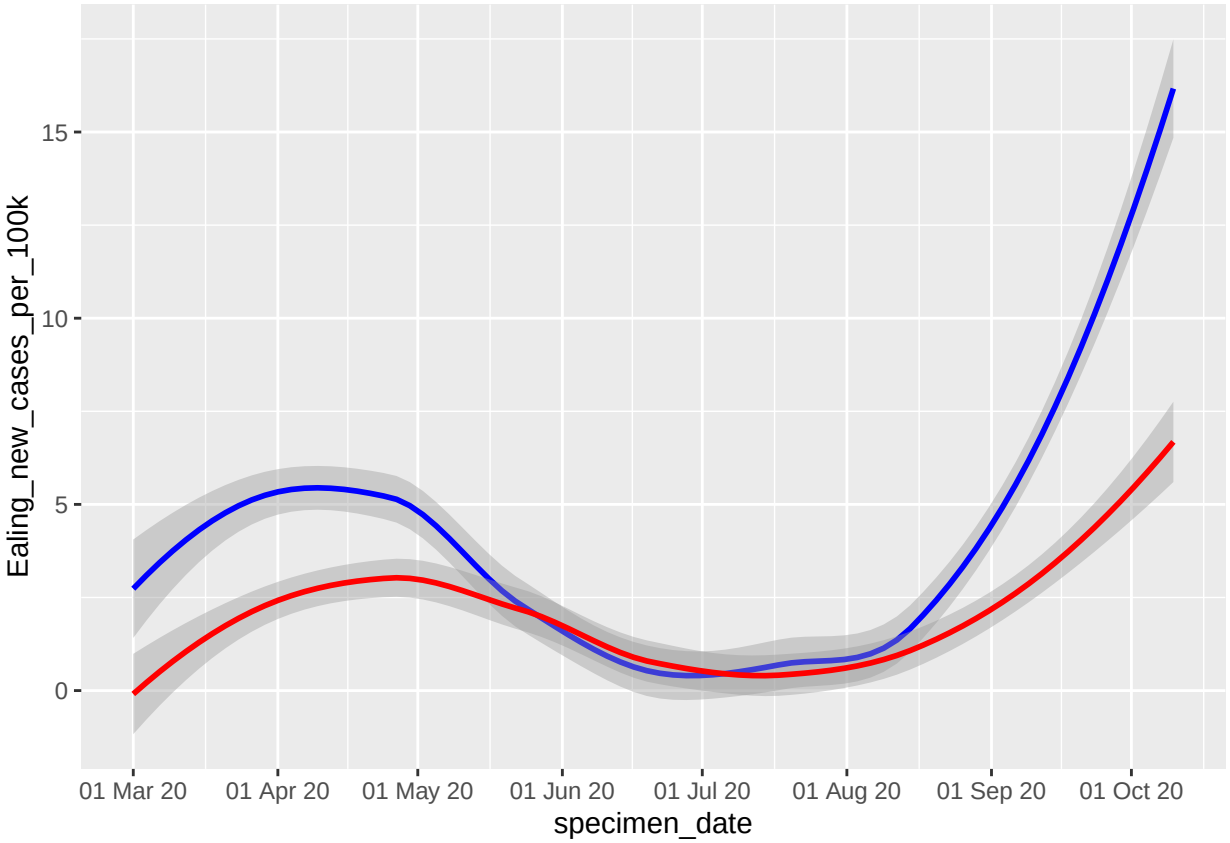
Now I can join `Ealing_covid_pop` to an identical table for Powys.

```
Ealing_v_Powys <- Ealing_covid_pop %>%
  full_join(Area_covid_pop("Powys"),
            by = c("specimen_date" = "specimen_date")
  )
```

In this table, columns ending in `.x` are referring to Ealing, and columns ending in `.y` are referring to Powys.

```
#Select specimen_date and the columns displaying new cases per 100,000 for each
#area.
Ealing_v_Powys_per_100k <- Ealing_v_Powys %>%
  select(specimen_date, new_cases_per_100k.x, new_cases_per_100k.y) %>%
  #Rename the columns to be clear what they represent
  rename(Ealing_new_cases_per_100k = new_cases_per_100k.x,
         Powys_new_cases_per_100k = new_cases_per_100k.y
         )

#Plot both new case rates per 100,000 people per day on the same chart
Ealing_v_Powys_per_100k %>%
  ggplot(aes(x = specimen_date)) +
  geom_smooth(aes(y = Ealing_new_cases_per_100k), col = 'blue') +
  geom_smooth(aes(y = Powys_new_cases_per_100k), col = 'red') +
  scale_x_date(date_breaks = "1 month", date_labels = "%d %b %y")
```



With the local authority data providing population statistics for each area, I calculated the daily new COVID-19 cases for Ealing per 100,000 of the population. The data shows that it was a very small percentage of the population getting infected each day, though the most recent figures showed that despite lockdown, almost 1% of the population had since the beginning of March tested positive for COVID-19. I used the daily new cases per 100,000 people figures to compare Ealing with a different region. I created a function to calculate new cases per 100,000 people per day for any area and compared the rate of new infection in Ealing to Powys, as Ealing is a densely populated urban area and Powys is a scarcely populated rural area. Figure 2 shows that both areas followed the same general trend, with cases rising from March into April, reaching a peak, decreasing during lockdown and levelling in the summer, only to begin to climb again from the beginning of August. Ealing's initial peak occurred earlier than Powys', but the rates were much higher, some days reaching above 12 per 100,000 whereas in Powys the figure only once went above 10. Both areas saw similar low figures during late June and through July, but from August to early October, Ealing has seen a much more drastic increase in cases, some days recording over 25 new cases per 100,000, and although Powys' cases are also rising, the gap between the two areas had not been bigger.