Visualizing COVID data in Ireland

Diksha Srivastava (21235117)

08/03/2022

Section 1: Cumulative number of COVID cases per 100,000 of population per county on 21 December 2021.

Reading the file

The dataset provided here contains the information on COVID in Ireland in 2020 and 2021. The dataset contains CountyName, Population, TimeStamp, DailyCCase, ConfirmedC, and geometry of each county in Ireland. Since the file is a shape (.shp) file, thus it contains the geometry column which contains the coordinates to generate the regions in the map.

I first downloaded the zip containing the dataset and then I unzipped it to a folder. The folder contains a .shp file which is read using st_read function.

```
library(sf)
library(dplyr)

# Read file
file <- "E:/Users/Diksha/Desktop/NUIG/DV/Statistics/CovidCountyStatisticsIreland_v2.shp"
# st_read function reads features from file or retrive their geometric type.
# quiet = TRUE suppresses the info on name, size, or multiple layers.
counties <- st_read(file, quiet = TRUE)
```

In order to use the data for visualization, I have first normalized the data. Normalization is used to convert the raw data to rates i.e., to adjust the data at different scales to a common reference scale. Here, we calculate it as Confirmed cases/Population * 100,000 for each county to find the cases per 100,000 of population.

In order to do this, I used dplyr's mutate functionality to add a new column named cases_per_100000 which is calculated using the formula shown above. I stored this data in a new variable named normalized_counties so that the previous/actual data 'counties' is never changed.

```
# Normalize data
normalized_counties <- counties %>%
  # adding the column containing cases per 100000
  mutate(cases_per_100000 = ConfirmedC/Population * 100000)
normalized_counties
```

```
## Simple feature collection with 17212 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -1182398 ymin: 6695905 xmax: -667637.5 ymax: 7449865
## Projected CRS: WGS 84 / Pseudo-Mercator
```

```
## First 10 features:
##    CountyName Population  TimeStamp DailyCCase ConfirmedC
## 1      Carlow      56932 2020-02-27         NA          0
## 2      Carlow      56932 2020-03-01          0          0
## 3      Carlow      56932 2020-03-02          0          0
## 4      Carlow      56932 2020-03-03          0          0
## 5      Carlow      56932 2020-03-04          0          0
## 6      Carlow      56932 2020-03-05          0          0
## 7      Carlow      56932 2020-03-06          0          0
## 8      Carlow      56932 2020-03-07          0          0
## 9      Carlow      56932 2020-03-08          0          0
## 10     Carlow      56932 2020-03-09          0          0
##                           geometry cases_per_100000
## 1  MULTIPOLYGON (((-749957.3 6...                0
## 2  MULTIPOLYGON (((-749957.3 6...                0
## 3  MULTIPOLYGON (((-749957.3 6...                0
## 4  MULTIPOLYGON (((-749957.3 6...                0
## 5  MULTIPOLYGON (((-749957.3 6...                0
## 6  MULTIPOLYGON (((-749957.3 6...                0
## 7  MULTIPOLYGON (((-749957.3 6...                0
## 8  MULTIPOLYGON (((-749957.3 6...                0
## 9  MULTIPOLYGON (((-749957.3 6...                0
## 10 MULTIPOLYGON (((-749957.3 6...                0
```

Since, for this visualization, we need cumulative cases per county on 21 December 2021, thus, I stored this data in a variable december_data_2021 by filtering only the data of 21 December 2021. Since each county will have only one data corresponding to the date "2021-12-21", thus, we get the cumulative cases for each county for this particular date. I only selected the two columns that we would need for this plot but since this is a sf object, it will contain the geometry column as well.

```
# Filter 21 December
december_data_2021 <- normalized_counties       %>%
  # filtering each county cases on 21-December-2021.
  filter(TimeStamp == "2021-12-21")             %>%
  # Selecting the columns required for this plot.
  select(CountyName, cases_per_100000)
december_data_2021
```

```
## Simple feature collection with 26 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -1182398 ymin: 6695905 xmax: -667637.5 ymax: 7449865
## Projected CRS: WGS 84 / Pseudo-Mercator
## First 10 features:
##    CountyName cases_per_100000                       geometry
## 1      Carlow         15406.10 MULTIPOLYGON (((-749957.3 6...
## 2       Cavan         15936.78 MULTIPOLYGON (((-884805.6 7...
## 3       Clare         11277.85 MULTIPOLYGON (((-1002912 70...
## 4        Cork         12090.78 MULTIPOLYGON (((-947249.9 6...
## 5     Donegal         18017.24 MULTIPOLYGON (((-806328.8 7...
## 6      Dublin         16360.82 MULTIPOLYGON (((-691660.8 7...
## 7      Galway         11986.84 MULTIPOLYGON (((-1006779 70...
## 8       Kerry         11356.94 MULTIPOLYGON (((-1142405 67...
```

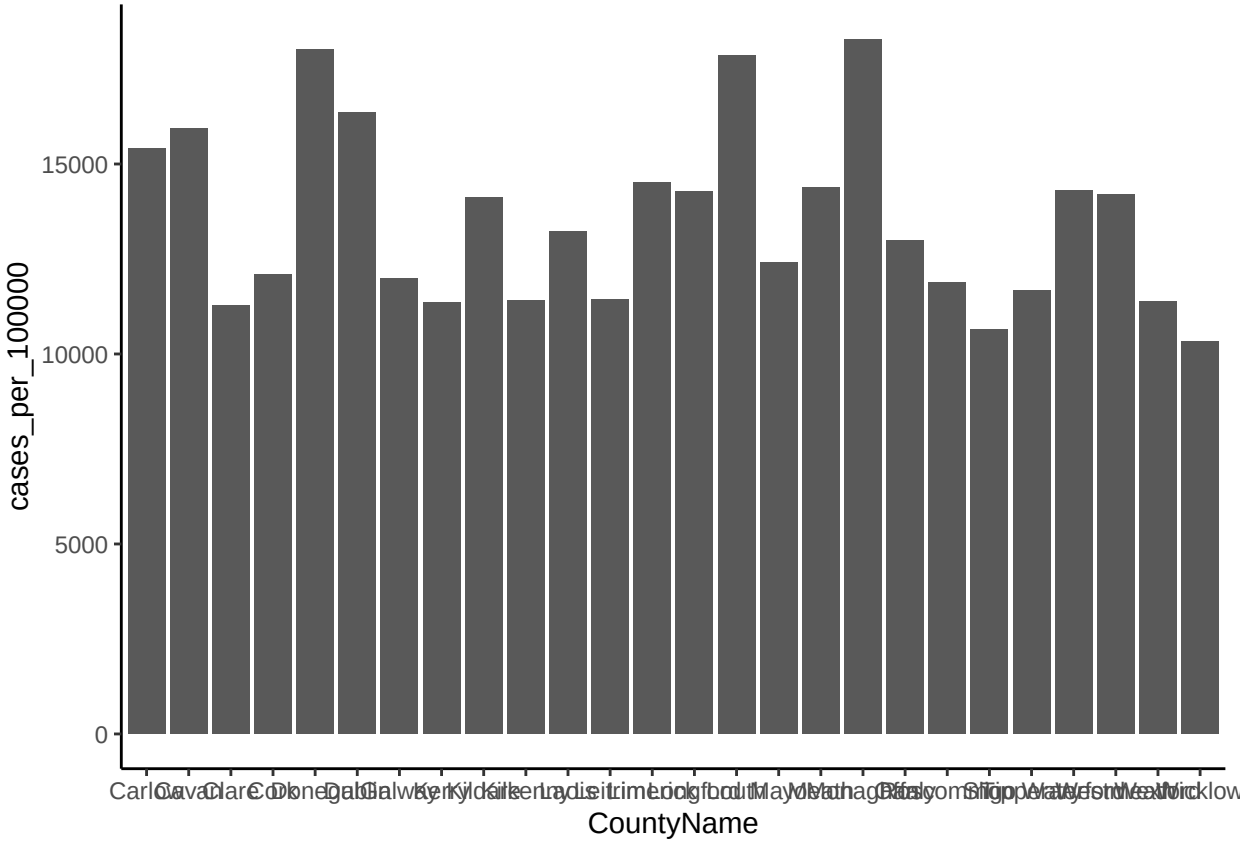
```
## 9     Kildare          14118.85 MULTIPOLYGON (((-760568.6 7...
## 10   Kilkenny          11413.66 MULTIPOLYGON (((-791492.6 6...
```

In order to visualize this dataset, I have used barplot. ggplot library provides us the means to create bar chart using geom_col() function. I have followed the following steps to get the final bar plot:

1. Create a basic barplot

Using the 21 December 2021 data created above, a bar plot is created with CountyName on the x-axis and cases per 100,000 population on y-axis. In order to remove gridlines from the background, I have used theme_classic() function as bar plot will be more clear to interpret.

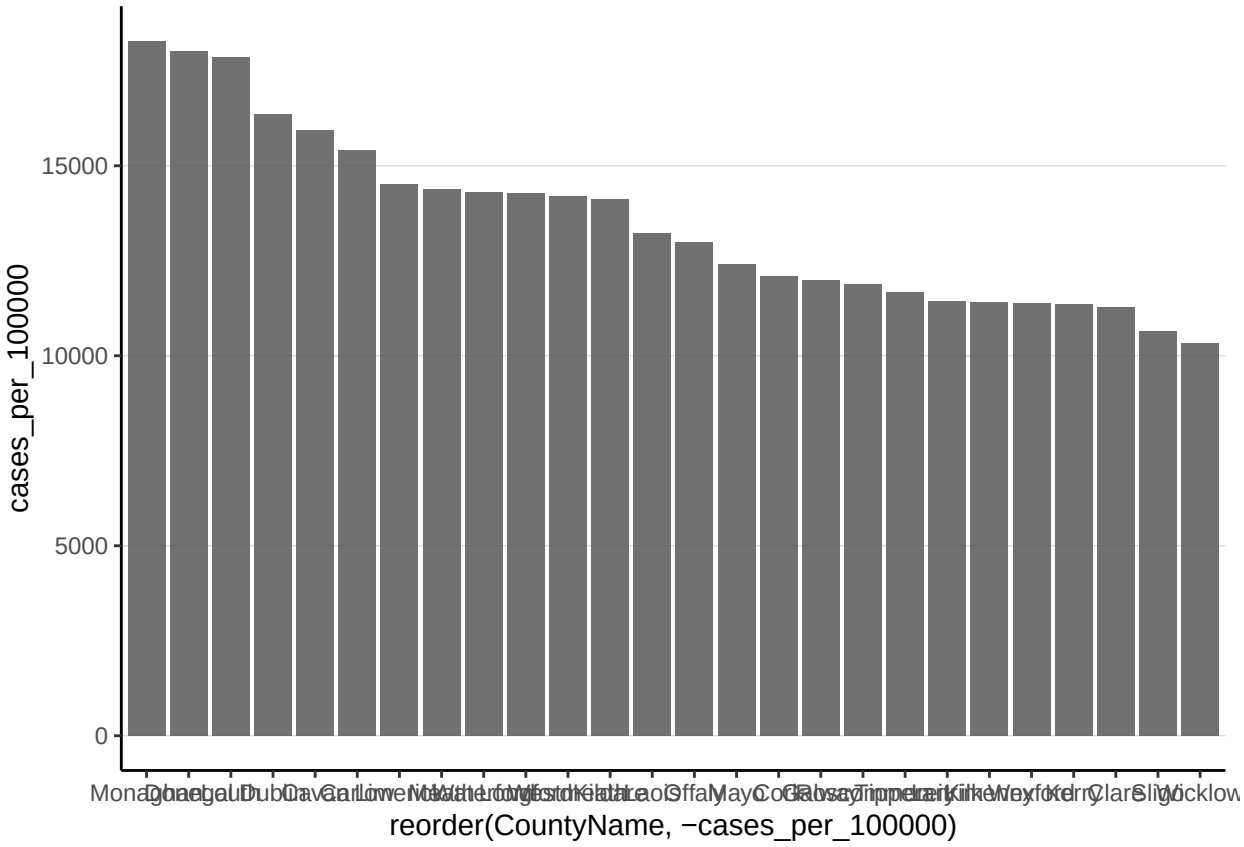
```
#basic bar plot
library(ggplot2)
# Setting the aesthetic (x-axis and y-axis) of the barplot.
ggplot(december_data_2021, aes(x = CountyName, y = cases_per_100000)) +
  # used for creating bar plot.
  geom_col()+
  theme_classic()
```



2. Ordering the plot

The plot will be more clear if we sort it according to the number of covid cases. Thus, I used reorder() function in the aesthetic to sort the plot according to the cases per 100,000 population. I ordered the plot in descending order, thus I had to use -cases_per_100000 as the default is ascending order.

```
ggplot(december_data_2021, aes(x = reorder(CountyName, -cases_per_100000),
                               y = cases_per_100000)) +
  # alpha values sets the transparency value of bars.
  geom_col(alpha = 0.85)+
  theme_classic() +
  # Setting the gridlines so that the bar is more interpretable.
  theme(panel.grid.major.y =
          # providing the settings of the gridline.
          element_line(size = 0.2,
                       linetype = 'solid',
                       colour = "lightgrey"))
```

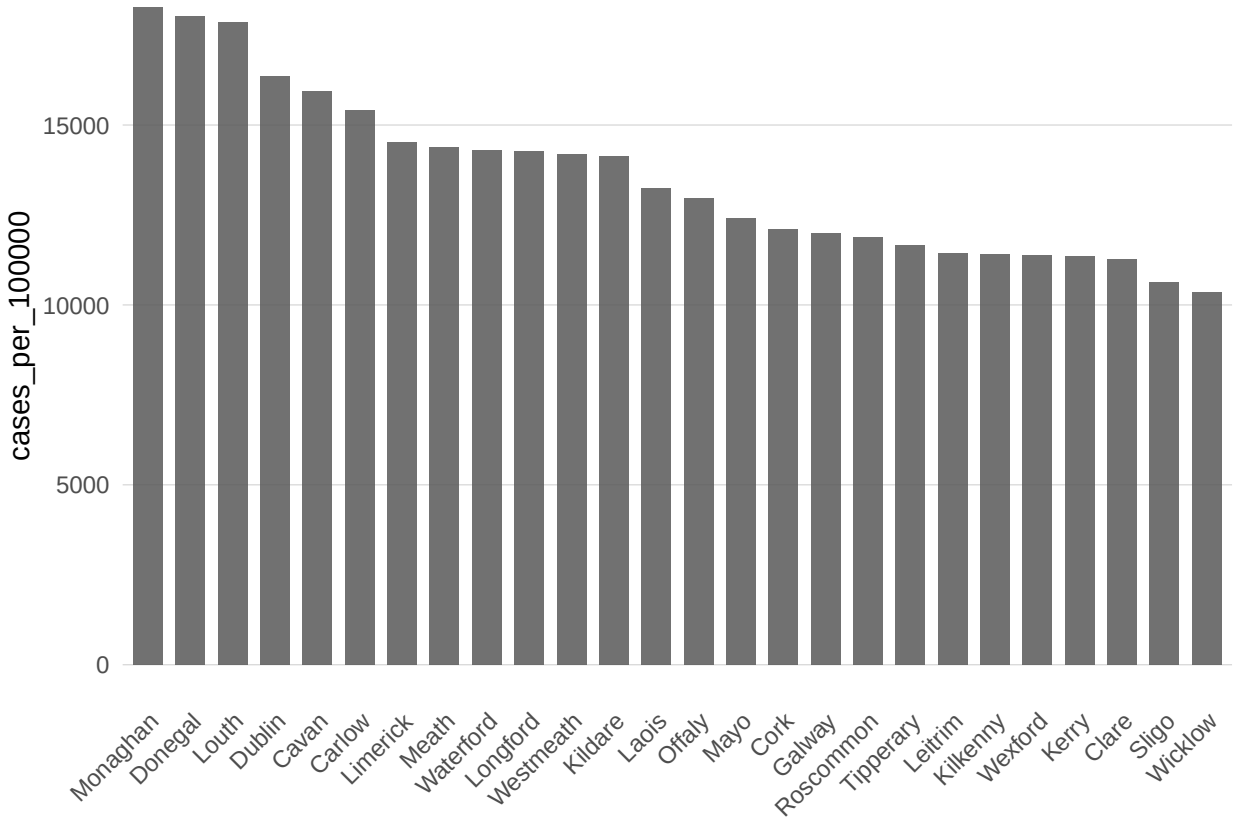


3. Modifying the X-axis labels.

Since the labels of x-axis are cluttered together, thus, using the theme we rotate the labels by 45 degree so that the labels are clearly visible and we bring the bars closer by setting their width to 0.7. The x-axis title has also been removed using the element_blank() function.

```
# Setting the county names horizontal
ggplot(december_data_2021, aes(x = reorder(CountyName, -cases_per_100000),
                               y = cases_per_100000)) +
  # setting the width of bars as 0.7.
  geom_col(alpha = 0.85, width = 0.7)+
  theme_classic() +
```

```
# removing the line, ticks, and the title of x-axis and y-axis.
theme(axis.line.x = element_blank(),
      axis.ticks.x = element_blank(),
      # setting the angle of the text to 45 degree so that the county names are visible.
      axis.text.x = element_text(angle = 45,
                                 vjust = 1,
                                 hjust = 1),
      axis.title.x = element_blank(),
      axis.ticks.y = element_blank(),
      axis.line.y = element_blank(),
      plot.margin = margin(3, 6, 3, 3),
      # setting the properties of the gridline.
      panel.grid.major.y =
               element_line(size = 0.2,
                     linetype = 'solid',
                     colour = "lightgrey"))
```

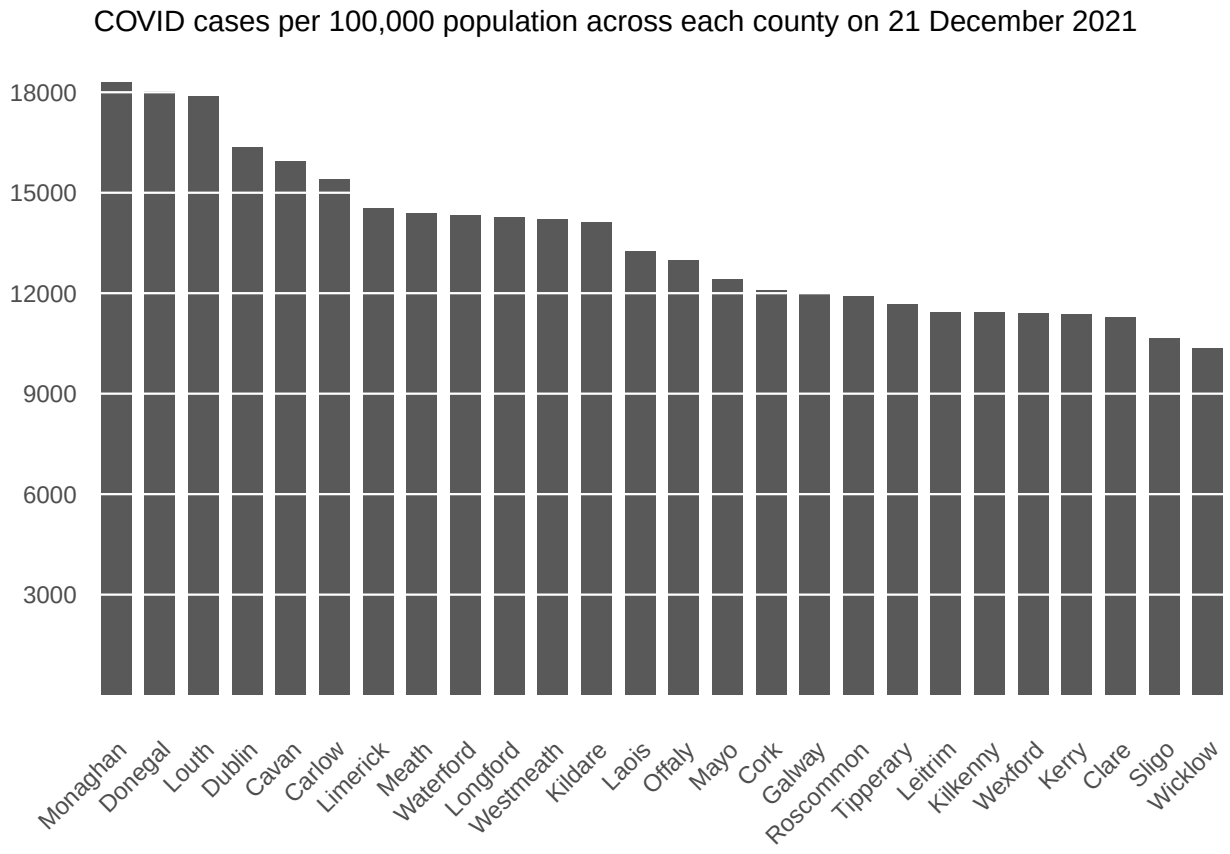


4. Setting the gridlines white

Since the y-axis is a continuous scale, thus we can set it's labels and breaks. I have taken this interval from 3000 to 19000 cases which ocurrs after every 3000 cases interval. All this is done inside scale_y_continuous() function. I have provided the breaks and labels to be set on the y-axis. I took the breaks as 3000 as with this break we can identify the value of each bar easily as there are six gridlines present using which we can quantify the cases per county. I have also made the gridlines white to reduce ink. Since now the gridlines will be part of overall background panel in ggplot, it is brought to the fore using panel.ontop = TRUE in the

theme function. Since the background panel also comes to the fore, we set it to blank using panel.background property. These white gridlines are visible only on the bars and helps in quantifying the difference between each bars. For example, it is clear that each county has atleast 9000 cases per 100,000. However, Monaghan, Donegal, and Louth has maximum number of cases with atleast 17,000 cases per 100,000.

```
ggplot(december_data_2021, aes(x = reorder(CountyName, -cases_per_100000),
                           y = cases_per_100000)) +
  # alpha value removed so that the white gridlines are visible clearly.
  geom_col(width = 0.7)+
  theme_classic() +
  # setting the breaks on y-axis.
  scale_y_continuous(breaks = seq(3000, 19000, by=3000),
    labels = as.character(seq(3000, 19000, by=3000))) +
  # setting the title of the ggplot.
  ggtitle("COVID cases per 100,000 population across each county on 21 December 2021") +
  theme(
    axis.title.y = element_blank(),
    axis.line.y = element_blank(),
    axis.ticks.y = element_blank(),
    axis.line.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.x = element_text(angle = 45,
                               vjust = 1, hjust = 1),
    axis.title.x = element_blank(),
    plot.title = element_text(hjust = 0.04, size = 11),
    plot.margin = margin(6, 6, 3, 3),
    # setting the background to be blank.
    panel.background = element_blank(),
    # setting the gridlines for y-axis.
    panel.grid.major.y =
      element_line(size = 0.4,
                   linetype = 'solid',
                   colour = "white"),
    # bringing the background to the fore.
    panel.ontop = TRUE)
```

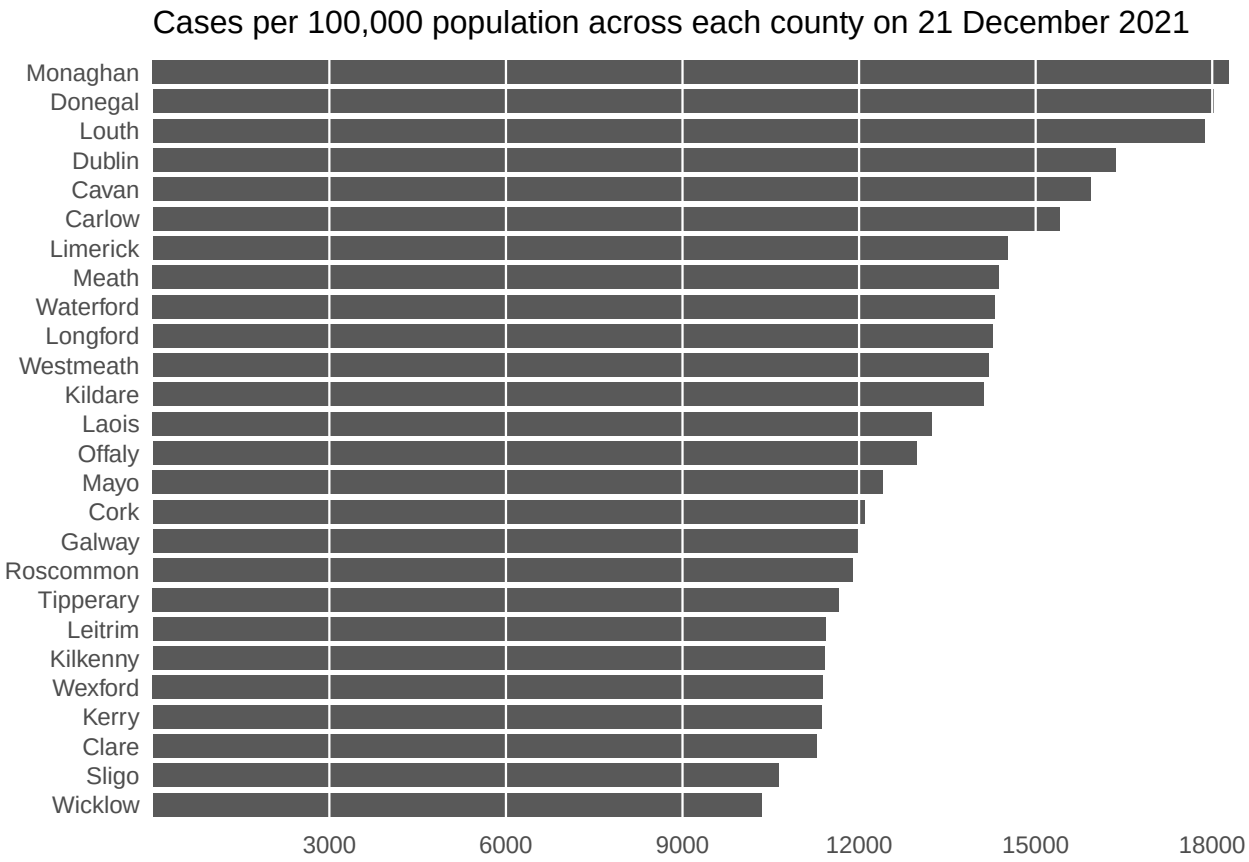


5. Making the bar graph horizontal

While the county names in the x-axis is clear, we still have to tilt our heads to read the name. Since the categories i.e., the county are more than 6 thus, it is preferrable to flip the x and y-axis, such that the bars are horizontal. The coord_flip() function is used to reverse the x-axis and y-axis. However, the original orientation of axes remains same. The white gridlines needs to be reversed as well. So here we provide the properties for the x-axis gridlines using panel.grid.major.x.

```
# Here we are ordering in ascending order of cases.
ggplot(december_data_2021, aes(x = reorder(CountyName, cases_per_100000),
                               y = cases_per_100000)) +
  geom_col(width = 0.8)+
  theme_classic() +
  scale_y_continuous(breaks = seq(3000, 19000, by=3000),
                     # keeps the bar closer to axes.
                     expand = c(0, 0),
                     labels = as.character(seq(3000, 19000, by=3000))) +
  ggtitle("Cases per 100,000 population across each county on 21 December 2021") +
  coord_flip(clip = "off") +
  theme(
    axis.title = element_blank(),
    axis.line.y = element_blank(),
    axis.ticks.y = element_blank(),
    axis.line.x = element_blank(),
    axis.ticks.x = element_blank(),
```

```
    axis.title.y = element_blank(),
    plot.title = element_text(size = 12),
    plot.margin = margin(3, 6, 3, 3),
    panel.background = element_blank(),
    panel.grid.major.x = element_line(size = 0.4,
                                      linetype = 'solid',
                                      colour = "white"),
    panel.ontop = TRUE)
```

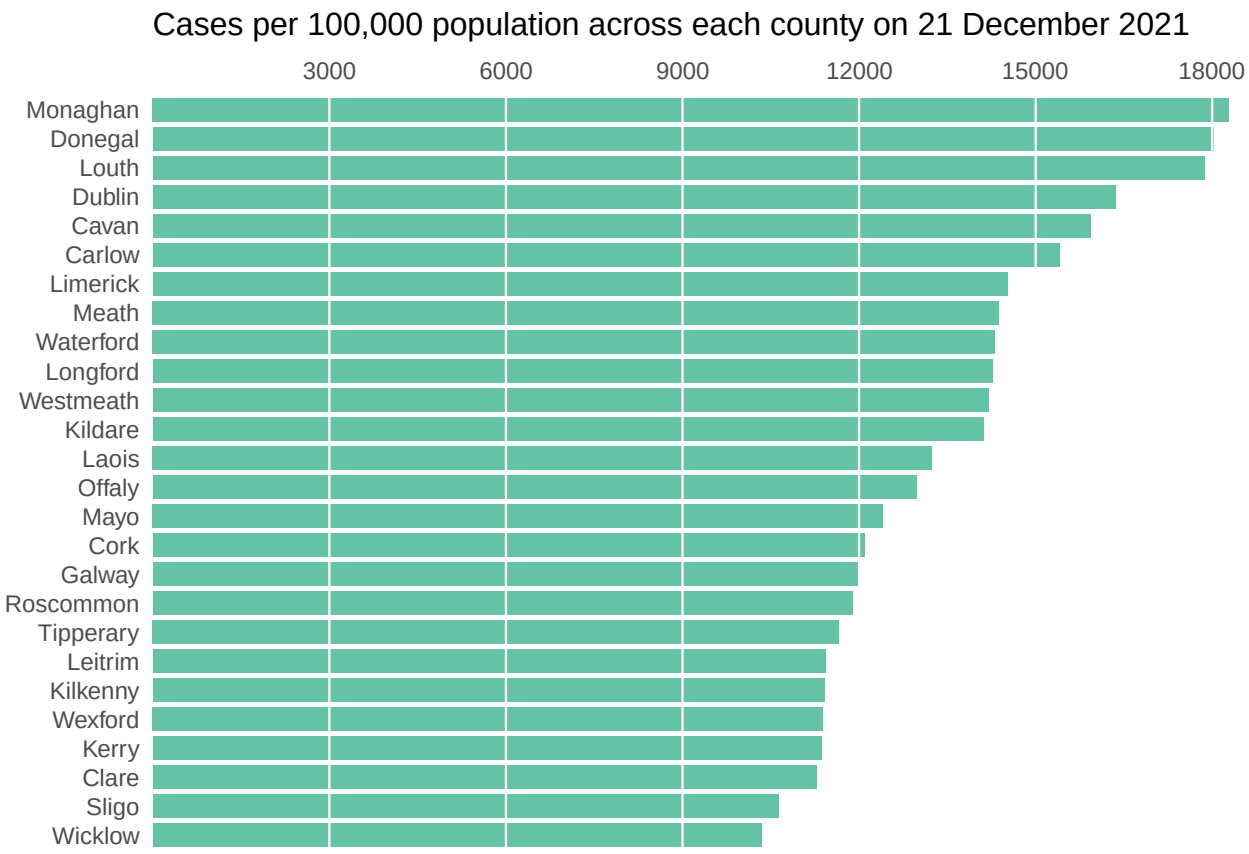


4. Final plot

In the final plot, the x-axis labels are put on top rather than bottom. It is done so because with the gridlines being white, it can become difficult to map the white gridlines with it's corresponding value on the x-axis at the bottom. Hence, we move the x-axis to the top. It is done by duplicating the x-axis using sec.axis = dup_axis() property inside scale_y_continuous. The x-axis at the bottom is set to blank in the theme using axis.title.x.bottom and axis.text.x.bottom. I have also filled the bar plot with a color using the fill property in geom_col(). Giving a color was not mandatory as we had only one categorical variable 'CountyName'. Grey color would have worked equally well. Had there been two categorical variables, the different colors would have helped in identifying different categories.

```
ggplot(december_data_2021, aes(x = reorder(CountyName, cases_per_100000),
                               y = cases_per_100000)) +
  geom_col(fill="#66c2a5", width = 0.8)+
  theme_classic() +
```

```
  scale_y_continuous(breaks = seq(3000, 19000, by=3000),
                     expand = c(0, 0),
                     labels = as.character(seq(3000, 19000, by=3000)),
                     sec.axis = dup_axis()) +
  ggtitle("Cases per 100,000 population across each county on 21 December 2021") +
  coord_flip(clip = "off") +
  theme(
    axis.title = element_blank(),
    axis.line.y = element_blank(),
    axis.ticks.y = element_blank(),
    axis.line.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    # setting the lower x-axis blank.
    axis.title.x.bottom =  element_blank(),
    axis.text.x.bottom =  element_blank(),
    plot.title = element_text(size = 12),
    plot.margin = margin(3, 6, 3, 3),
    panel.background = element_blank(),
    panel.grid.major.x = element_line(size = 0.4,
                                      linetype = 'solid',
                                      colour = "white"),
    panel.ontop = TRUE)
```



1. The above bar plot is suitable to show cumulative number of cases per 100,000 population across each

county on 21 December 2021. Bar plot is used in this case because, it is easier for human eyes to identify the 'length' aesthetic in comparison to area or any other aestehtic.

2. The breaks used on the x-axis i.e., from 3000 to 19000 with a 3000 interval is neither large nor less. A less break interval could have caused a cluttered plot and would not have been clear and the humans could get distracted from so many lines. An interval of 3000 ensures that the gridlines are not in large quantity and also helps in quantifying the bar values. For example, we can say that the number of cases per 100,000 in cork is nearly around 12,000. Thus, we get an approximate quantity for each county.

3. The gridlines helps in comparing different counties as well. For example, we could say that the number of cases per 100,000 in county Cork and county Carlow differed by approximately 3500 cases. Thus, we can say that county Carlow was more affected by COVID than county Cork. It is also clear that Monaghan was the most affected county in Ireland with approximately 18,500 cases per 100,000 and Wicklow was the least affected county from COVID with approximately 10,500 cases per 100,000.

4. Since all the visual elements in this graph is horizontal, thus it is much easier to read.

Section 2: A Visualization that allows reader to read how each county differs from the mean cumulative number of cases per 100,000 in the country as of 21 December 2021.

In order to find how each county differes from the mean cumulative number of cases per 100,000, we first find the mean number of cases on 21 December 2021. Since in the previous data we already have the cumulative number of cases for each county, we can find the mean from that data. I have made use of mean() function to find the mean of cases per 100,000. I have set na.rm = TRUE such that if any NA values are encountered then they are handled and does not affect in finding mean.

```
mean_cases <-  mean(december_data_2021$cases_per_100000, na.rm = TRUE)
mean_cases
```

```
## [1] 13529
```

We see that the mean number of cases obtained is 13529 per 100,000.

To show how each county differs from the mean, I will make use of Diverging Bar chart. A diverging bar chart handles positive and negative values. This graph shows the divergence of cases from a reference value which can be either 0 or a mean value. For our case, we would show the divergence from the mean value.

Thus, we will first create a dataset that contains the difference from the mean value. The previous data we had contained cases per 100,000 for each county, we will use that. We will mutate a new column named diff_from_mean which stores the difference of cases in each county from the mean value. We will also mutate a column named pos which contains boolean value TRUE or FALSE. TRUE is stored in pos if the difference calculated is a positive value else FALSE is stored. This pos column is used for setting the color-coding when the graph will be plotted for positive and negative values.

```
data_diverge <- december_data_2021                          %>%
  mutate(diff_from_mean = cases_per_100000 - mean_cases) %>%
  # check if the difference obtained is positive or negative.
  mutate(pos = diff_from_mean >= 0)
data_diverge
```

```
## Simple feature collection with 26 features and 4 fields
```

```
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -1182398 ymin: 6695905 xmax: -667637.5 ymax: 7449865
## Projected CRS: WGS 84 / Pseudo-Mercator
## First 10 features:
##    CountyName cases_per_100000                       geometry diff_from_mean
## 1      Carlow         15406.10 MULTIPOLYGON (((-749957.3 6...       1877.097
## 2       Cavan         15936.78 MULTIPOLYGON (((-884805.6 7...       2407.777
## 3       Clare         11277.85 MULTIPOLYGON (((-1002912 70...      -2251.154
## 4        Cork         12090.78 MULTIPOLYGON (((-947249.9 6...      -1438.217
## 5     Donegal         18017.24 MULTIPOLYGON (((-806328.8 7...       4488.236
## 6      Dublin         16360.82 MULTIPOLYGON (((-691660.8 7...       2831.820
## 7      Galway         11986.84 MULTIPOLYGON (((-1006779 70...      -1542.161
## 8       Kerry         11356.94 MULTIPOLYGON (((-1142405 67...      -2172.058
## 9     Kildare         14118.85 MULTIPOLYGON (((-760568.6 7...        589.846
## 10   Kilkenny         11413.66 MULTIPOLYGON (((-791492.6 6...      -2115.344
##      pos
## 1   TRUE
## 2   TRUE
## 3  FALSE
## 4  FALSE
## 5   TRUE
## 6   TRUE
## 7  FALSE
## 8  FALSE
## 9   TRUE
## 10 FALSE
```

I have included the colorblindr library to fetch two colors which will be used to fill bars for positive and negative values.

```
library(colorblindr)
cbPallete <- palette_OkabeIto
# choosing the 5th and 6th color of the pallete.
cbTwoColors <-cbPallete[c(5, 6)]
```

The diverging bar chart diverges from 0 where 0 represents the mean number of cases. If the bars had positive value that means that the cases for that county were more than the mean and if the bars had negative values, then that county had less cases than the mean. To plot this, I have kept the x-axis as CountyName and y-axis as difference from the mean calculated. The graph is ordered using reorder() function based on the difference from mean value. The graph is ordered in ascending way.

The property position = "identity" helped in removing the warning about the stacking not well-defined for negative values.

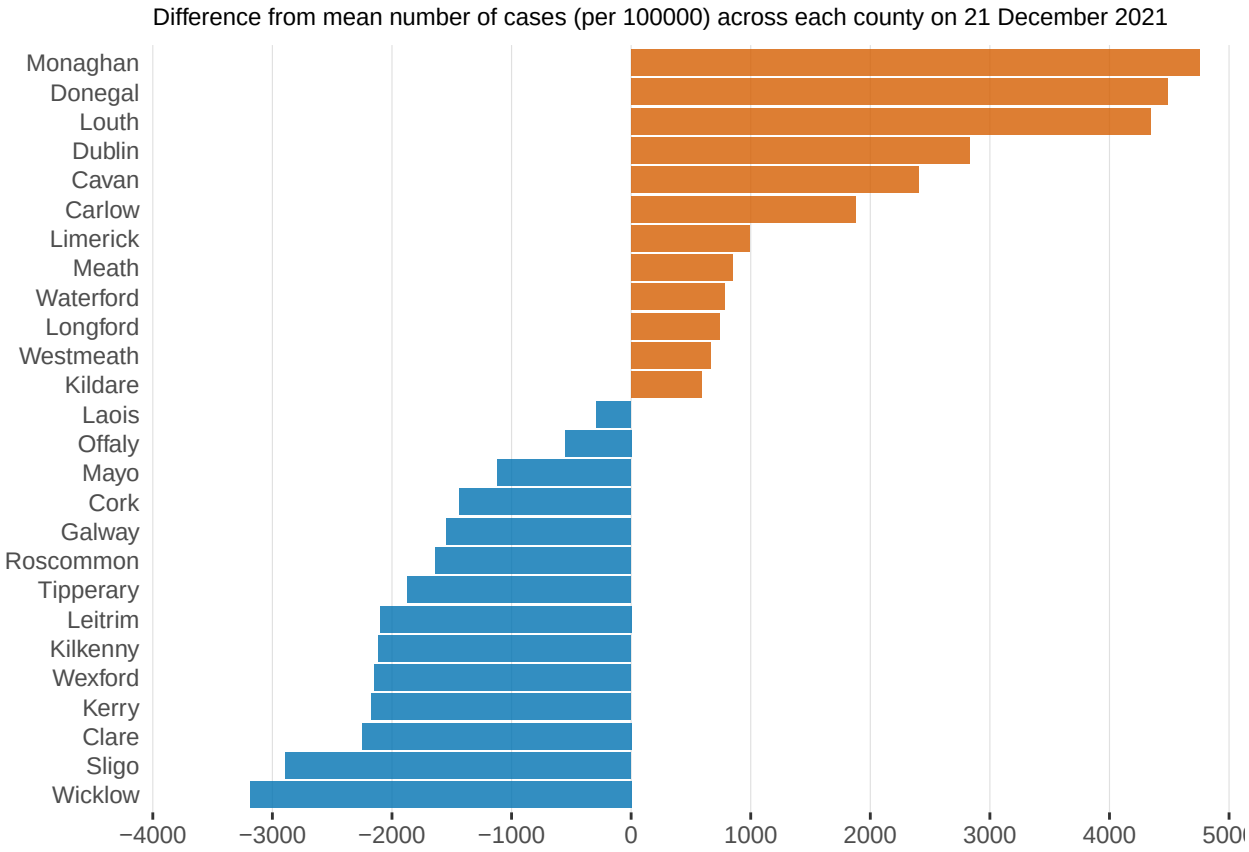
I have also flipped the plot as number of categories are more, hence a horizontal graph would be much clear to interpret. I have set the limits and breaks from -4000 to 5000 with an interval of 1000. These intervals helps in identifying the quantitaive difference from mean. For example, from the plot it is clear that county 'Cavan' differed from the mean cases by approximately 2400 number of cases per 100,000.

I have also manually filled the colors selected to the bar plot using scale_fill_manual() function. I have taken the values such that the positive values are represented by orange color and negative values by blue color. It is so because in this case if a county has negative value from mean that means they had less number of cases as compared to other counties. Hence, it is a good thing and should be represented by a color showing positive vibe i.e., blue. However, if a county had positive value then that means they had more cases from

the mean value and hence, it is a danger sign. Danger signs are usually shown in red shades, hence, orange is suitable to show positive value.

The gridlines are set using panel.grid.major.x property. A 'Tufte' approach is not used as it is easier to interpret this graph without any cluttering.

```
ggplot(data_diverge, aes(x = reorder(CountyName, diff_from_mean),
                         y = diff_from_mean, fill = pos)) +
  geom_col(position = "identity", alpha = 0.8) +
  theme_classic() +
  coord_flip(clip = "off") +
  ggtitle("Difference from mean number of cases (per 100000) across each county on 21 December 2021") +
  scale_y_continuous(limits = c(-4000, 5000),
                     breaks = seq(-4000, 5000, by= 1000) ,
                     # distance from the axes.
                     expand=c(0,0),
                     labels = as.character(seq(-4000, 5000, by= 1000))) +
  scale_fill_manual(values = cbTwoColors) +
  theme(
    axis.line.y = element_blank(),
    axis.ticks.y = element_blank(),
    axis.line.x = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    plot.margin = margin(6, 6, 3, 3),
    panel.grid.major.x = element_line(size = 0.2, linetype = 'solid',colour = "grey88"),
    plot.title = element_text(size = 9),
    # removing the legend from the plot.
    legend.position = "none"

  )
```



From the above plot, it is clear that county Monaghan had approximately 4700 more number of cases from mean and county Wicklow had approximately 3100 less number of cases from mean. Thus, Monaghan was the most affected county from COVID and Wicklow was the least affected one. All measure are in cases per 100,000 of population.

Section 3: Choropleth Visualisation

A choreopleth visualization can help in representing spatial variations of cumulative number of cases across Ireland. A choreopleth graph is created using geom_sf() function but before moving forward, we first need to create a common scale to represent the map from the year 2020 and 2021.

We already have the dataset for 21 December 2021, so we will only filter out the cumulative cases for 21 December 2020.

```
december_data_2020 <- normalized_counties %>%
  filter(TimeStamp == "2020-12-21")       %>%
  select(CountyName, cases_per_100000)
```

In order to represent the map on a common-scale, we have to discretize continuous data into several intervals. These intervals will be mapped to discrete colors which will be shown in legend. I will use discrete color scales because it is easier to discriminate these colors when plotted on a map, and hence the map would be more interpretable.

To discretize continuous scale, we will perform the following steps:

1. Since we need to create a common scale, thus the minimum value of the plot will be obtained from 2020 data and the maximum value will be obtained from the 2021 data as 2020 observed less cases and 2021 observed more cases.

```
# discretizing data
library(plyr)
# using plyr to round to nearest minimum 1000.
scale_minimum<-round_any(min(december_data_2020$cases_per_100000), 1000, f = floor)
# using plyr to round to nearest maximum 1000.
scale_maximum<- round_any(max(december_data_2021$cases_per_100000), 1000, f = ceiling)
```

2. Discretize the continuous data into equal intervals. I have created breaks which start from 0 and run to 20,000 with an interval of 2000. Since the minimum value was 0 so I didn't subtract any value from it. To the maximum value I added 2000 so that the last interval is not too small. It will be easier to represent it as a legend. I have created 10 intervals as it will be easier to represent 10 distinct colors on the map. Any value above it would make it difficult to differentiate colors. Any value lower than 10 will make it difficult to distinguish between areas affected differently by COVID.

With the help of cut() function, we discretize the continuous data i.e., cases_per_100000 into 10 intervals and keep it in column cases_per_100000_D for both dataset.

```
breaks<-seq(scale_minimum,scale_maximum+2000, by =2000)

december_data_2021$cases_per_100000_D <- cut(december_data_2021$cases_per_100000,
                                             breaks = breaks,
                                             dig.lab = 5)

december_data_2020$cases_per_100000_D <- cut(december_data_2020$cases_per_100000,
                                             breaks = breaks,
                                             dig.lab = 5)
```

3. Discretize the continuous color scale into 10 intervals so that each interval have different colors. We are able to find nlevels i.e., number of colors from December 2021 data as it has all the intervals. The colors are obtained from hcl.colors() function which creates n contiguous colors and we have used Inferno color scale to visualize our data.

```
library(colorspace)

# number of colors is equal to the number of intervals in the data.
nlevels<- nlevels(december_data_2021$cases_per_100000_D )
# This returns nlevels number of colors i.e., 10 colors in this case.
pal <- hcl.colors(nlevels, "Inferno", rev = TRUE)
# reducing the brightness of the color.
pal_desat<-desaturate(pal,amount = 0.2)
# Creating custom labels like (0k - 2k]
lbl <- breaks/1000
# creating label names based on number of levels/intervals calculated.
labs_plot <- paste0("(", lbl[1:nlevels], "k-", lbl[1:nlevels+1], "k]")
labs_plot
```

```
##  [1] "(0k-2k]"   "(2k-4k]"   "(4k-6k]"   "(6k-8k]"   "(8k-10k]"  "(10k-12k]"
##  [7] "(12k-14k]" "(14k-16k]" "(16k-18k]" "(18k-20k]"
```

4. Create the choropleth maps for 2020 and 2021 data using the discretized data and colors. We can use scale_fill_manual function to specify the mapping of colors. The map is created using geom_sf() function. geom_sf() creates different geometrical objects based on geographic regions made up of polygons. geom_sf() assumes that there is a default column named geometry which is present in our data. It reads the geometry column which represents the coordinates of each county. Since we have assigned discrete colors to the intervals, a low/light color indicates that the area has less number of cases and a bright/dark color indicates that the area has more number of cases.

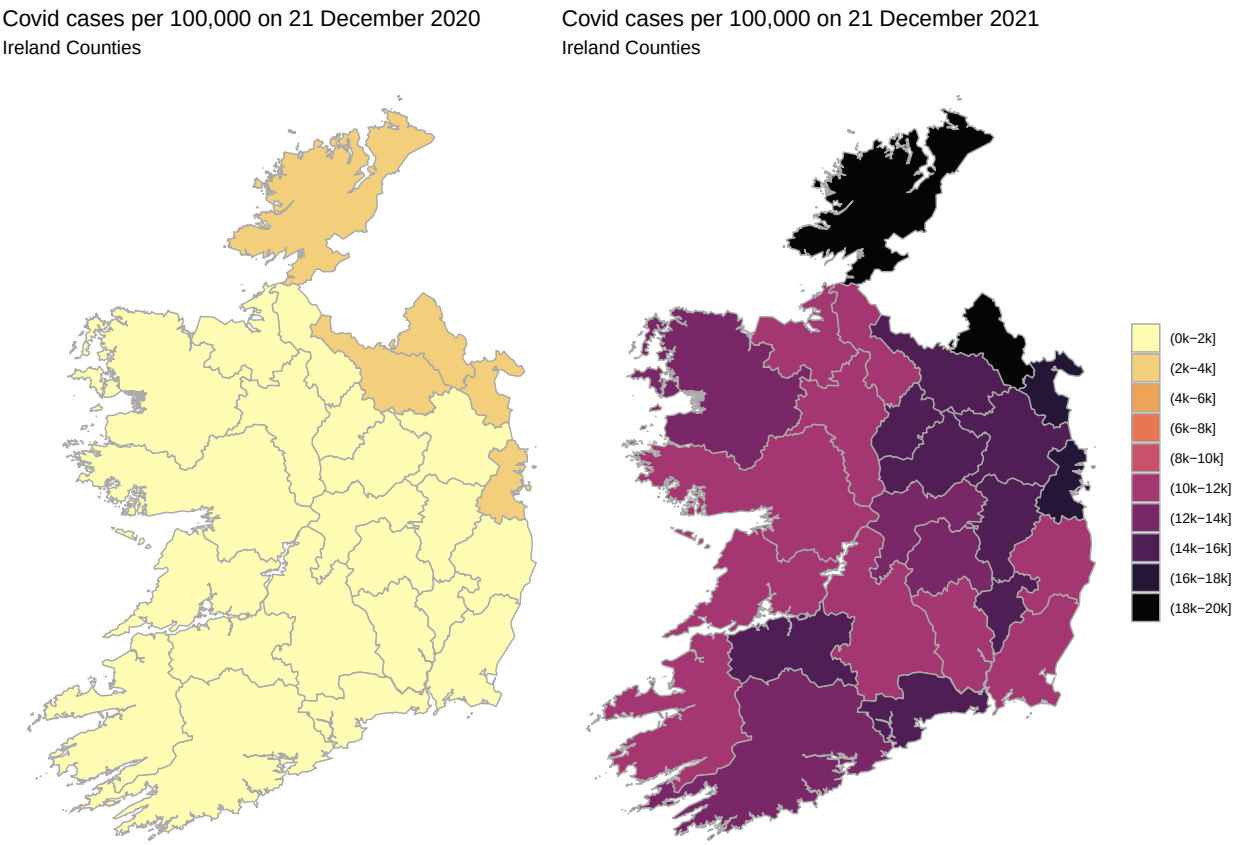
In order to show the common legend, I have created a legend on the right side of graph and on the first graph the legend is hidden. The legend is positioned vertically and towards right.

```
choropleth_2021 <- ggplot(december_data_2021) +
  # filling the colors based on the intervals obtained.
  geom_sf(aes(fill = cases_per_100000_D),
          color = "darkgrey",
          linetype = 1,
          lwd = 0.4) +
  # setting the title and subtitle of the plot.
  labs(title = "Covid cases per 100,000 on 21 December 2021",
       subtitle = "Ireland Counties") +
  # setting the custom pallete.
  scale_fill_manual(values = pal_desat,
                    # does not remove the unused factor levels from the scale.
                    drop = FALSE,
                    na.value = "grey80",
                    # custom labels set.
                    label = labs_plot,
                    # position of legend is set as right and the direction set is vertical.
                    guide = guide_legend(direction = "vertical",
                                         # it will tell to use 10 rows for the legend.
                                         nrow = 10,
                                         label.position = "right")) +
  # Theme
  theme_void() +
  theme(legend.title = element_blank(),
        # we keep the size of text of legend as 8.
        legend.text = element_text(size=8),
        # setting the caption of the plot.
        plot.caption = element_text(size = 7, face = "italic"))

# storing the map of 2020 data.
choropleth_2020 <- ggplot(december_data_2020) +
  geom_sf(aes(fill = cases_per_100000_D),
          color = "darkgrey",
          linetype = 1,
          lwd = 0.4) +
  # Labs
  labs(title = "Covid cases per 100,000 on 21 December 2020",
       subtitle = "Ireland Counties") +
  # setting the custom pallete.
  scale_fill_manual(values = pal_desat,
                    drop = FALSE,
                    na.value = "grey80") +
  # Theme
```

```
  theme_void() +
  theme(legend.title = element_blank(),
        legend.text = element_blank(),
        legend.position = "none",
        legend.key.height = grid::unit(0.4, "cm"),
        plot.caption = element_text(size = 7, face = "italic"))
```

The patchwork library can be used to merge two plots. With the help of + sign multiple ggplot graphics can be overlayed on one another. Thus, it is used to merge both the choropleth plots and showing them side-by-side.

```
library(patchwork)
choropleth_2020 + choropleth_2021
```



Discretizing the data and assigning colors to each interval helps in identifying the patterns in the data. In this case, we can say that overall the number of cases in 2020 were comparatively less than the cases in 2021. It is also prominent that there were some counties which were more affected than other areas in both years. These counties are Donegal, Monaghan, Louth, Dublin. In 2020, they had approximately 2k to 4k cases per 100,000. However, in 2021, the cases drastically increased to 16k to 20k cases. Overall the number of cases in 2021 remained above 8k, however, overall the number of cases in 2020 remained below 4k. There are many counties that had cases between 8k-10k in 2021, however, they had 0k-2k cases only in 2020. Thus these counties saw a major increase in the number of cases. These counties are Galway, Claire, Sligo, Wicklow, Wexford and some more. We don't see any cases lying in range 4k-8k in any year.

Section 4: Time series bar graph of daily number of confirmed cases along with a line representing 7-day average for a particular period.

I will be visualizing time series bar chart for county 'Galway' for a period of 6 months i.e., from 1 July 2021 to 21 December 2021.

Along with the time series bar chart we have to plot a line as well representing 7-day average, we will first create the data that has all these informtaion.

To calculate the 7-day average for the period of 6 months, I have used the method provided in the worksheet to calculate the average. This method takes in the date, value for which average needs to be calculated, average period like 7-day/20-day, and center which plots the average at the centre of the time window. This method calculates the mean of values for the given range. For example, if 7-day average is required, then the mean values will be calculated from current_date - (7/2) to current_date + (7/2). This is done using the offset and range values.

```
# The moving average function which calculates the mean for the given range.
moving_ave <- function(date, value, range, center = TRUE) { # This code was developed by Claus Wilke
  if (isTRUE(center)) {
    offset <- ceiling(range/2)
  } else {
    offset <- range
  }
  vapply(
    1:length(value),
    function(i, date, value) {
      focal_day <- date[i]
      first_day <- focal_day - offset + 1
      last_day <- focal_day - offset + range
      idx <- date >= first_day & date <= last_day
      if (head(date, 1L) > first_day || tail(date, 1L) < last_day) {
        NA_real_
      } else {
        mean(value[idx])
      }
    },
    double(1),
    date,
    value
  )
}
```

With the above logic of moving average, we cannot find the average for beginning and end dates. I was able to get the average for the beginning dates. I applied the following logic. Since the period that I am choosing

starts from 2021-07-01 and we know that moving average will calculate average from current_date - 3 day i.e., 2021-06-28, so I included dates from 2021-06-28 and passed it to moving average function. Now the moving average was able to find the average for 2021-07-01 but not for 2021-06-28. Since we only needed the plot from 2021-07-01 so I filtered out the data and used it to plot the graph. I couldn't do the same for december data since the data after 21 December 2021 is not available.

We are using the lubridate library as it is easier to work with dates using this package.

```
startdate<-'2021-07-01'
enddate<-'2021-12-21'
time_series_mov_ave <- counties                                                    %>%
  # filtering out the data for Galway county from 28 June 2021 to 21 December 2021.
  filter(CountyName == 'Galway' & TimeStamp >= '2021-06-28' &
         TimeStamp <= enddate)                                                     %>%
  # selecting ony the date and daily number of cases required for this case.
  select(TimeStamp, DailyCCase)                                                    %>%
  # calling the moving average function to calculates the 7-day average for each date
  # and mutating the result in the column close_7d_ave.
  mutate(close_7d_ave = moving_ave(TimeStamp, DailyCCase, 7, center = TRUE))%>%
  # filtering out the data from 01 July 2021 so that data from June are removed.
  filter(TimeStamp >= startdate)

time_series_mov_ave
```

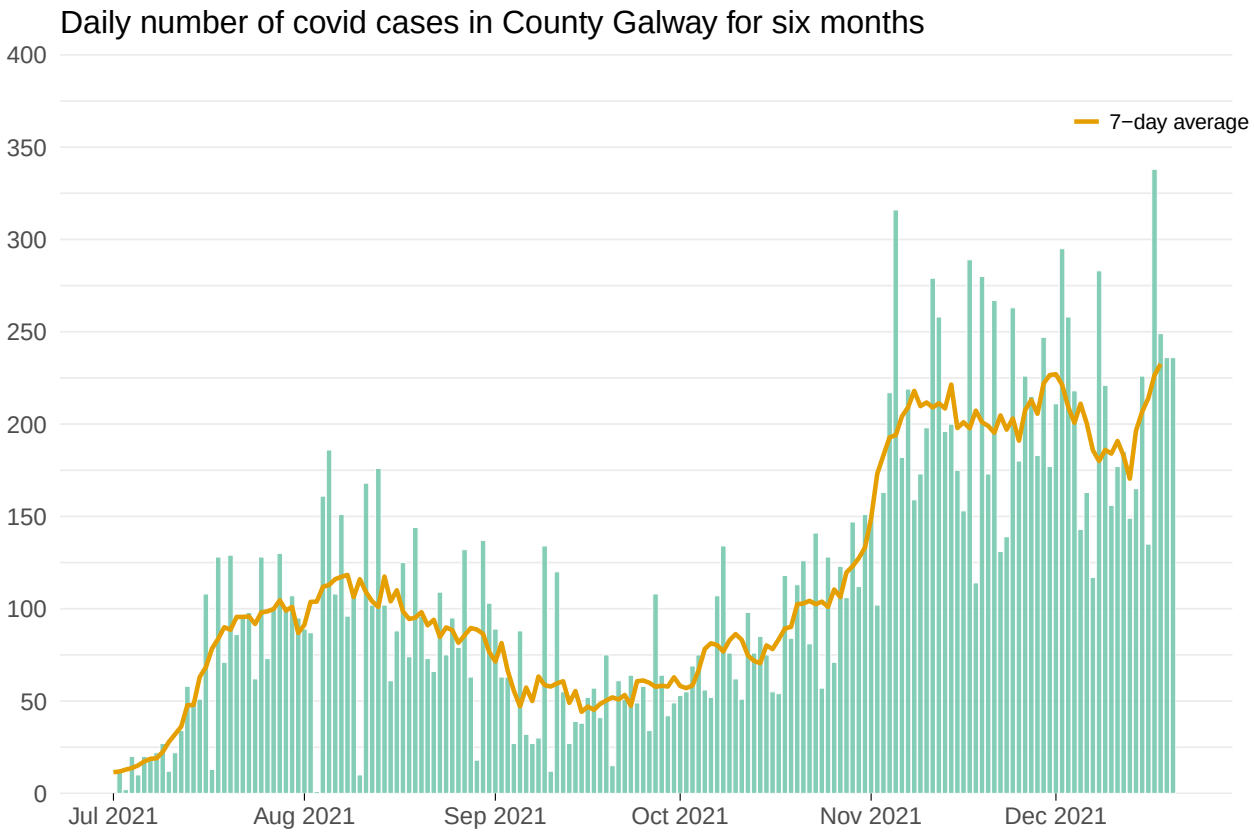
```
## Simple feature collection with 174 features and 3 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -1146896 ymin: 6977123 xmax: -887052.1 ymax: 7117096
## Projected CRS: WGS 84 / Pseudo-Mercator
## First 10 features:
##     TimeStamp DailyCCase                       geometry close_7d_ave
## 1  2021-07-01         12 MULTIPOLYGON (((-1006779 70...     11.42857
## 2  2021-07-02         12 MULTIPOLYGON (((-1006779 70...     11.85714
## 3  2021-07-03          2 MULTIPOLYGON (((-1006779 70...     12.85714
## 4  2021-07-04         20 MULTIPOLYGON (((-1006779 70...     13.71429
## 5  2021-07-05         10 MULTIPOLYGON (((-1006779 70...     15.14286
## 6  2021-07-06         20 MULTIPOLYGON (((-1006779 70...     17.28571
## 7  2021-07-07         20 MULTIPOLYGON (((-1006779 70...     18.71429
## 8  2021-07-08         22 MULTIPOLYGON (((-1006779 70...     19.00000
## 9  2021-07-09         27 MULTIPOLYGON (((-1006779 70...     22.42857
## 10 2021-07-10         12 MULTIPOLYGON (((-1006779 70...     27.85714
```

First, I am creating a bar chart using geom_col() function such that x-axis has dates for 6 months and y-axis has daily number of cases for county Galway. Then, we also plot line for the entire period using geom_line() such that x axis is same as date, and y-axis has 7-day average daily number of cases.

To color the line, I have used the color from OkabeIto pallete in scale_color_manual() property. I have also provided the label for the line drawn but I have removed any title using name= NULL property.

Using the scale_y_continuous(), I have set the limits from 0 to 400 cases with a break of 50 intervals. Using the scale_x_date(), I have provided the format of dates and intervals. The limit of dates is from 1 July 2021 to 21 December 2021. The ymd(startdate) signifies the format of date used i.e., year-month-date. Since I am plotting the data for 6 months, I have kept the break as 1 month interval. I have set the format of labels using date_format("%b%Y") which is present in 'scales' library which signifies that the format of label will be short-form of month like Aug for August and the complete year i.e., 2021.

```
library(colorblindr)
library(lubridate)
library(scales)

ggplot(time_series_mov_ave,
            aes(x=TimeStamp, y=DailyCCase)) +
  # setting the size of bar 0.05 as a lot of data needs to be plotted.
  geom_col(fill="#66c2a5",alpha=0.8, colour="white", size=0.05)  +
  # using na.rm to handle NA values.
  geom_line(aes(TimeStamp, close_7d_ave, color = "7d"), size = 0.8, na.rm = TRUE) +
  scale_color_manual(
    values = c(
      `7d` = palette_OkabeIto[1]
    ),
    labels = c("7-day average"),
    name = NULL
  ) +
  scale_y_continuous(limits = c(0, 400),
                     breaks = seq(0, 400, by= 50),
                     expand = c(0, 0)) +
  scale_x_date(limits = c(ymd(startdate), ymd(enddate)),
               breaks = "1 month",
               labels=date_format("%b %Y")) +

  ggtitle("Daily number of covid cases in County Galway for six months")+
  theme_minimal() +
  theme(
    legend.just = c(0, 1),
    legend.position = c(0.85,0.97),
    # providing the size of text of legend.
    legend.text = element_text(size=8),
    legend.title = element_blank(),
    # setting the size of the key i.e., line in legend
    legend.key.size = unit(0.8,"line"),
    legend.spacing.x = unit(0.1, 'cm'),
    # setting the gridlines of x-axis as blank.
    panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank(),
    # setting the gridlines for y-axis to identify the number of cases.
    panel.grid.major.y = element_line(size=0.3),
    panel.grid.minor.y = element_line(size=0.3),
    axis.ticks.x =element_line(size=0.2),
    # removing the title and ticks for x-axis and title for y-axis.
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    plot.margin = margin(14, 7, 3, 1.5),
    plot.title = element_text(size = 12))
```



The above time series plot helps in identifying the trends for daily number of cases in county Galway for a period of 6 months. It is evident that overall the cases increased from July 2021 to December 2021. However, we can also notice a slight downfall in the number of cases for the month September and October. This change is evident from the line graph that we plotted. Using the bar graph, we can identify that the maximum number of cases occurred in December with approximately 330 cases.

While the line graph helps in visualizing the overall trend, it still is not smooth enough.

Section 5: Time series line graph

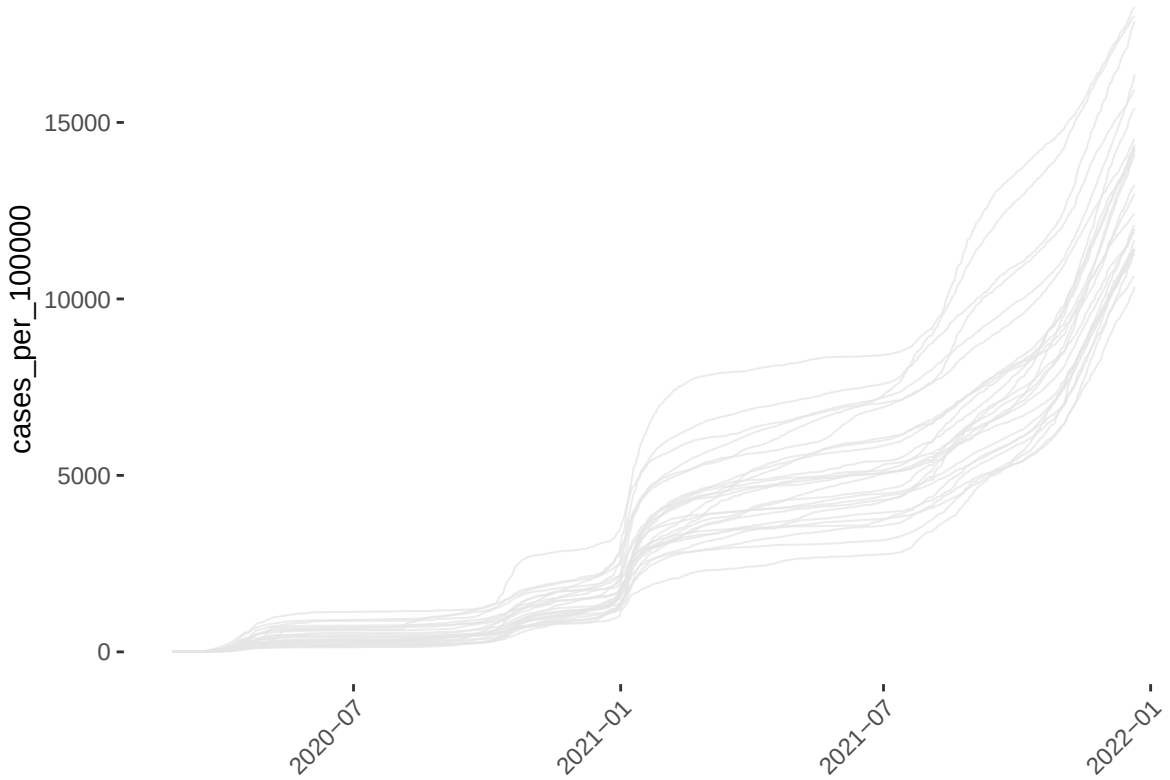
First, we identify the counties that needs to be plotted on the foreground. The foreground will have Galway, county with minimum cases, and county with maximum cases. Since we have cumulative number of cases, thus, we can have total number of cases in each county as of 21 December 2021. Thus, we first filter out the data of 21 December 2021. Then we find the counties with maximum and minimum number of cases. Once that is found we keep it in a variable to be used further.

```
cumulative_county <- normalized_counties %>% filter(TimeStamp == "2021-12-21")
max_county <- cumulative_county[which.max(cumulative_county$cases_per_100000),1]$CountyName
min_county <- cumulative_county[which.min(cumulative_county$cases_per_100000),1]$CountyName

foreground <- c("Galway", max_county, min_county)
foreground
```

```
## [1] "Galway"   "Monaghan" "Wicklow"
```

Since, we want to plot the cumulative cases for each county and highlight the ones that have minimum and maximum cases, so we will divide the plot into two. One will be the background layer and one will be the foreground layer. Background layer will contain cases of all counties and foreground will only contain the counties that we want to highlight. First we plot the background data in grey color and we will keep it's alpha value and size less so that it is pale enough so that the foreground counties will be visible clearly. We do all this using geom_line() function.

```
# Plotting background layer.
# plotting timestamp on x-axis and cases per 100,000 on y-axis.
p_nations<- ggplot(normalized_counties, aes(x =TimeStamp, y=cases_per_100000)) +
  # grouping the data by county and keeping the size and alpha value of less less.
  geom_line(aes(group = CountyName),size= 0.35, na.rm = TRUE, color="grey90", alpha =0.75,
            show.legend = FALSE ) +
  # this theme makes the background white
  theme(panel.grid.major = element_blank(),
        panel.background = element_blank(),
        axis.line = element_blank(),
        axis.title.x=element_blank(),
        # keeping the text of x axis title at 45 degree.
        axis.text.x = element_text(angle = 45,
                                   vjust = 1, hjust = 1),
        legend.key = element_rect(fill = NA, colour = NA),
        plot.margin = margin(14, 14, 8, 14),
        plot.title = element_text(size = 12))

p_nations
```



Next, we plot the foreground data with distinct colors so that they are easy to identify. We first select the subset of data containing the data for foreground counties only.

```
# fetching foreground data.
foreground_counties<- subset(normalized_counties, CountyName %in% foreground)
```

Then, we plot the foreground counties such that the size of line and alpha value is greater than the background data. On the x-axis, we take TimeStamp and on the y-axis we take cumulative cases per 100,000 and group them by each county.

The scale_colour_manual() helps us to set different colors for foreground data.

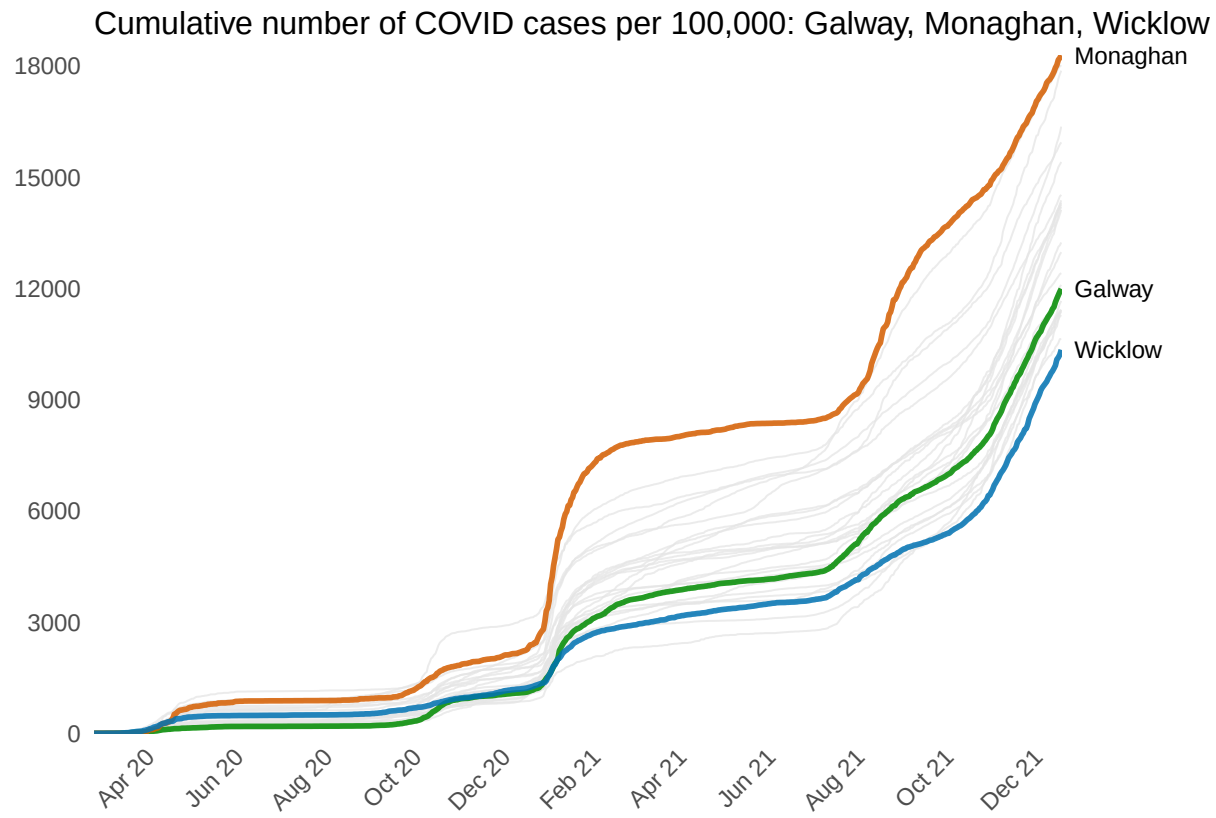
scale_y_continuous() helps us set the intervals and a duplicate axis. We are using duplicate y-axis so that we don't have to use legend and the graph is more easy to interpret. The labels and breaks are set from 0 to 19,000 with an interval of 3000 so that we don't have many points on the y-axis which can make the graph look cluttered. We set the expand property to 0,0 so that the line is more closer to the axes.

We use sec.axis property to set the duplicate axis. This duplicate axis is required so that we don't have to use legend to show the foreground data and the county name of foreground data can be shown where the line ends. All the other counties are hidden. For the duplicate axis also we require breaks and labels. Thus, we set the breaks using foreground_counties_final$cases_per_100000. foreground_counties_final variable contains the final cumulative case which is obtained by fetching the data of 21 December 2021. This data will be the same where the line graph ends, hence, we can show the county names here using labels property and we get our final graph.

We set the x-axis labels as well such that it is rotated 45 degrees using axis.text.x in theme and we set an interval of two months. We format the date using date_format() to show only month and year in the labels.

```
# getting the cumulative count on 21 December 2021 to be plotted on duplicate y-axis.
foreground_counties_final <- filter(foreground_counties,
                                    TimeStamp == ymd("2021-12-21"))
#Your foreground layer
p_nations3 <- p_nations +
  geom_line(data=foreground_counties, size =1, alpha=0.85,
            show.legend = TRUE,
            (aes(x =TimeStamp, y=cases_per_100000, colour= CountyName,
                 group = CountyName))) +
  scale_colour_manual(values = c("green4","#D55E00", "#0072b2")) +
  scale_y_continuous(labels = seq(from = 0, to =19000, by=3000),
                     breaks =seq(from = 0, to =19000, by=3000),
                     expand=c(0,0),
                     # creating duplicate axis.
                     sec.axis = dup_axis(
    breaks = foreground_counties_final$cases_per_100000,
    labels = foreground_counties_final$CountyName) )+
  # setting the title of the graph
  ggtitle("Cumulative number of COVID cases per 100,000: Galway, Monaghan, Wicklow") +
  scale_x_date(breaks = "2 month", labels=date_format("%b %y"), expand=c(0,0) ) +
  theme(
    legend.position = "none",
    axis.ticks.y.right = element_blank(),
    axis.ticks.y = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y= element_blank(),
    axis.text.x = element_text(angle = 45,
                               vjust = 1, hjust = 1),
```

```
    axis.text.y.right = element_text(colour="black", size =9),
    # setting the size of title.
    plot.title = element_text(size = 12))
p_nations3
```



The final plot shows the COVID cases of Galway in comparison to county with highest (Monaghan) cases and lowest (Wicklow) cases. It also shows the cumulative cases across each county in background data. It is eminent that the number of cases for each county increased from 2020 to 2021. Galway had comparatively less number of cases as compared to other counties as it is much closer to Wicklow in the graph which had minimum number of cases. Monaghan on the other hand, had high number of cases as compared to other counties since the beginning. Monaghan also saw a fall in the cases during August 2021 but in the later months, the cases increased.