Explainer

I have added explainers for each data piece along with links to the source code.

# News Story

There is no up-to-date dataset of the number of MPs who have registered for proxy votes or who these votes are assigned to. In order to create this dataset, I built a [scraper using Python](https://github.com/tomsaunders98/tomsaunders98.github.io/blob/master/dataproject/NewsArticle/main.py) and Selenium which scraped through the order roll of each day Parliament was sitting. I then searched for a list of MPs who had requested proxy votes and added them to a basic spreadsheet. To work out each MP’s party I used the Parliament Data API.

I then turned to [R to wrangle and visualise my data](https://github.com/tomsaunders98/tomsaunders98.github.io/blob/master/dataproject/NewsArticle/Viz.R). Firstly, I had to do some extensive manipulation of the data in order to remove duplicated data (caused by a change in the way proxy votes were presented on Hansard) and to convert boolean values into a tally for each party. To visualise the graph I used ggplot with the geom\_ribbon function which allowed me to visualise the split between each party in how they used proxy votes. I also used Cairo along with the package ggthemes to give the graph a more professional look and to rasterise the ribbons so that the lines look smooth.

To calculate how many MPs were assigned to each party whip I had to add extra functions to my scraping algorithm which registered each MPs proxy counterpart.

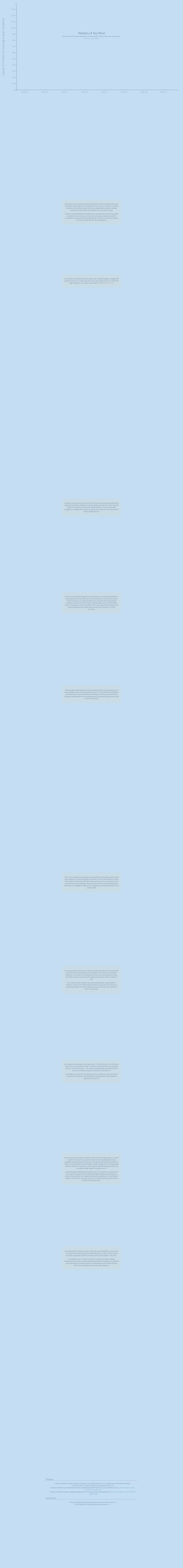
**Word Count: 215**

# 

# Feature Story

The data for this piece was sourced from the ONS along with the University College London COVID-19 Social Study. To visualise this piece I used [d3js](https://github.com/tomsaunders98/tomsaunders98.github.io/blob/master/dataproject/js/main.js), [CSS](https://github.com/tomsaunders98/tomsaunders98.github.io/blob/master/dataproject/css/style.css), [HTML](https://github.com/tomsaunders98/tomsaunders98.github.io/blob/master/dataproject/index.html) and a small bit of Jquery.

The page is built dynamically each time it is loaded, using the viewport size. The central chart is positioned to be fixed in the page whilst the description boxes are positioned down the page so that when you scroll the data points are revealed at the same time as their associated descriptions. A birds-eye view of the first part of this web page (flipped horizontally) looks like this:



The gradual scroll-reveal of the line graph was done by calculating how many data points were in the central graph, then building the page so that a scroll of one-fifth of the screen size reveals one data point. The graph animations were built by manipulating the stroke-dasharray and stroke-dashoffset properties of the line graph to give the effect that the graph is drawing itself across the page.

Once the end of the page is reached the graph is transformed from a fixed position to a static one, allowing the conclusion to be visible below.

**Word Count: 196**