A perculiar way in which the UK's constituency-based electoral system shapes media coverage is that the names of certain towns/districts have an outsized effect. For instance, in the 2019 UK general election, much was made of Workington Man* in Cumbria- a seat that had fairly consistently returned Labour MPs in the modern era.

One particular media trend made possible by the variety of names for UK seats is to alliterate between constituencies that are seen as showing a range of geography/opinion/etc. This is best summed up in a great exchange between the absolute boy, and Health Secretary, Matt Hancock, and Kay Burley:

On Sky, Matt Hanock says new cancer treatments are being rolled out "from Barnsley to Bassetlaw; from Wigan to Warrington."Kay Burley: "That's not very far, you know."Hancock: "It's also happening in Cornwall."

— Peter Walker (@peterwalker99) 30 October 2019

Given that I had an afternoon off sick from work, and enjoy wasting my time on such things, I wanted to see what the best constituencies to use for 'From Xx to Xy' in British politics is. For this I'm going to use mostly data that is hosted on this website, however, where it isn't I've made it pretty clear in comments where it can be downlaoded.

*for a good take on this, see here

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\rob-getty\Desktop\cleanup desktop\geo_data\boundary\Data\GB", layer: "westminster_
## with 632 features
## It has 15 fields
## Integer64 fields read as strings:  NUMBER NUMBER0 POLYGON_ID UNIT_ID
```

First, and easiest, let's start with geographic distances between constituencies. For this I use the Ordnance Survey boundary line dataset which gives shapefile of each constituency in the UK.

After some string regex to match names between datasets, I also removed all constituencies beginning with North/South/East/West (as 'From East Surry to East Hampshire' doesn't really have a ring to it) and also only took seats within England or Wales (more on why later).

We're then left with 350 (out of 650 total) seats which we can plot, filled by the first letter of their name.

```
#load tidyverse for munging
library(tidyverse)

#this data can be found at https://www.ordnancesurvey.co.uk/business-government/products/boundaryline
#open as:
# constituency_shapefiles <- readOGR(dsn = "where/you/downloaded",
#                                    layer = "westminster_const_region")
#using rgdal and sf for geospatial work
library(rgdal)
library(sf)

constituency_geography <- constituency_shapefiles %>%
  st_as_sf() %>%
  #some munging to line up datasets
  mutate(name = gsub(" Co Const| Burgh Const| Boro Const", "", NAME)) %>%
  mutate(name = case_when(
    grepl("London and Westminster", name) ~ "Cities of London and Westminster",
    grepl("-.*-", name) ~ gsub("(-)([a-z]{1})(.*-)", perl = TRUE, "\\1\\U\\2\\E\\3", name),
    grepl("St\\. ", name) ~ gsub("St\\. ", "St ", name),
    grepl(" of ", name) ~ gsub(" of ", " Of ", name),
```

```
    grepl("Newcastle upon ", name) ~ gsub(" upon ", " Upon ", name),
    TRUE ~ name
  )) %>%
  #get the first letter
  #removing compass directions
  filter(!grepl("North |East |South |West ", name) & !grepl(" .* ", name)) %>%
  mutate(first_letter = gsub("(.)(.*)", "\\1",  name)) %>%
  #only going to play with English constituencies here
  filter(grepl("^E|^W", CODE)) %>%
  #remove Chorley (speaker's seat)
  filter(!grepl("Chorley", name)) %>%
  #select and rename relevant columns
  select(WSTid = CODE, WSTnm = name, first_letter)

constituency_names <- constituency_geography %>%
  `st_geometry<-`(NULL)

#ggthemes for map theme
library(ggthemes)

#plot remaining constituencies coloured based on first letter
first_letter_plot <- ggplot() +
  geom_sf(data = constituency_geography, aes(fill = first_letter)) +
  scale_fill_discrete(guide = FALSE) +
  theme_map()

#plot(first_letter_plot)
```

To calculate the distance between any two constituencies, I use the center location of each, calculated using sf::st_centroid(). Grouping by first letter then creating a matrix from each to each is simple enough using sf::st_distance() as follows:

```
#get the coordinates of the center of each constituency
geographic_centers <- constituency_geography %>%
  st_centroid() %>%
  split(f = .$first_letter)

#function to find distances between center points
get_distances <- function(letter_list) {
  constituencies <- letter_list$WSTnm
  first_letter <- unique(letter_list$first_letter)

  #get distance to/from every center point with same first letter
  distance_matrix <- st_distance(letter_list, letter_list)

  distances_df <- distance_matrix %>%
    as.data.frame()
  names(distances_df) <- constituencies

  melted_df <- distances_df %>%
    pivot_longer(., names(.), names_to = "to", values_to = "distance") %>%
    mutate(from = rep(unique(to), each = length(unique(to)))) %>%
    mutate(first_letter = first_letter)
```

```
  return(melted_df)
}

#run the function to get the distances between constituencies with same
#first letter
constituency_interdistances <- map_df(geographic_centers, get_distances)
```

We can then find the longest distance (in metres) between the centre of constituencies, grouped by the first letter of their name using some simple muning:

```
#find the longest distances
longest_distances <- constituency_interdistances %>%
  #arrange by longest first
  arrange(-distance) %>%
  #take the longest per first letter
  filter(!duplicated(first_letter)) %>%
  filter(distance != 0)

#show the ongest 10 interdistances
head(longest_distances %>% arrange(-distance))
```

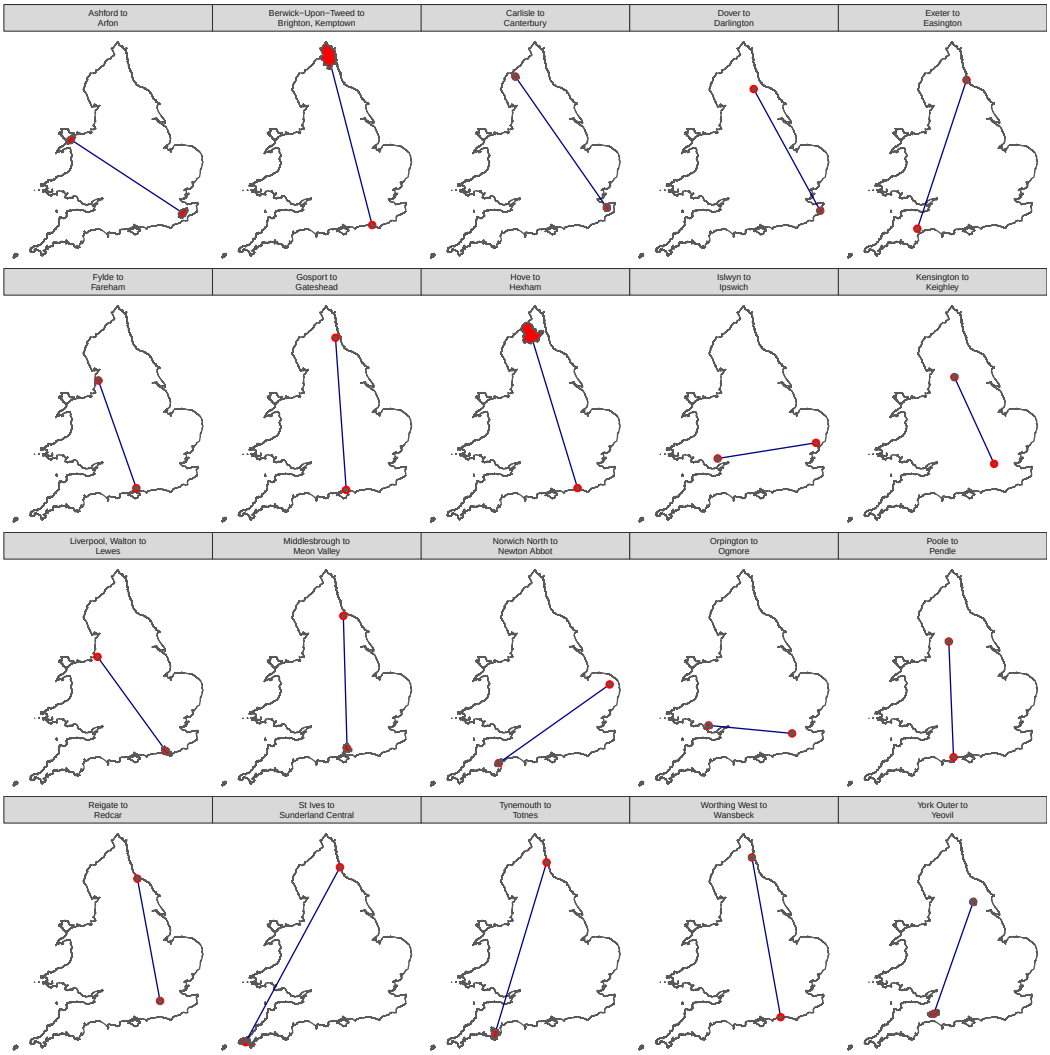
```
## # A tibble: 6 x 4
##   to                 distance from               first_letter
##   <chr>                 <dbl> <chr>              <chr>
## 1 St Ives             601117. Sunderland Central S
## 2 Tynemouth           540605. Totnes             T
## 3 Berwick-Upon-Tweed  528160. Brighton, Kemptown B
## 4 Worthing West       490544. Wansbeck           W
## 5 Hove                488181. Hexham             H
## 6 Carlisle            483127. Canterbury         C
```

Perhaps not surprisingly, St Ives (in the far South West on England) to Sunderland Central (in the far North East) is the furthest distance (601km). We can see though that there's a fair few first letter for which we have a pair of constituencies that are pretty far away from each other.

To plot the longest distance between a pair of constituencies that alliterate is simple enough. I also load a shapefile of the outline of England and Wales to pretty up the plots and create lines between each constituency. Where constituencies are too small to be plotted on this scale, I use a red dot.

```
#shapefile of England and Wales for plotting
eng_wal <- "C:/Users/rob-getty/Desktop/netlify_blog/static/files/constituency_distances/england_wales_sl
  readRDS()

#filter the longest journey per letter
selected_constituencies <- constituency_geography %>%
  filter(WSTnm %in% pivot_longer(longest_distances, cols = c("to", "from"))$value) %>%
  left_join(.,
            longest_distances %>%
              mutate(journey = paste(to, from, sep = " to\n")) %>%
              select(first_letter, journey),
            by = "first_letter")
```

```r
#get the center coordinates of constituencies
#to help plotting small constituencies
plotting_points <- do.call(rbind, geographic_centers) %>%
  filter(WSTnm %in% selected_constituencies$WSTnm) %>%
  left_join(.,
            longest_distances %>%
              mutate(journey = paste(to, from, sep = " to\n")) %>%
              select(first_letter, journey),
            by = "first_letter") %>%
  st_transform(crs = st_crs(eng_wal))

#calculate straight lines between two constituencies
plotting_lines <- plotting_points %>%
  split(f = .$journey) %>%
  map_df(., function(data) {
    coords <- rbind(st_coordinates(data[1,]), st_coordinates(data[2,]))
    line <- st_linestring(coords)
    df <- st_sfc(line, crs = st_crs("+init=epsg:27700")) %>%
      as.data.frame() %>%
      mutate(journey = unique(data$journey))
  }) %>%
  st_as_sf(crs = st_crs(plotting_points))

#plot the longest journey between constituencies with the same first letter
alliterative_journeys_plot <-  ggplot() +
  geom_sf(data = eng_wal, fill = "white") +
  geom_sf(data = plotting_lines, colour = "darkblue") +
  #some constituencies are too small to plot as shapefiles
  geom_sf(data = plotting_points, colour = "red", size = 2.5) +
  geom_sf(data = selected_constituencies, fill = "red") +
  theme_map() +
  #split by first letter
  facet_wrap(~journey)

plot(alliterative_journeys_plot)
```


Ashford to
Arfon
Berwick−Upon−Tweed to
Brighton, Kemptown
Carlisle to
Canterbury
Dover to
Darlington
Exeter to
Easington
Fylde to
Fareham
Gosport to
Gateshead
Hove to
Hexham
Islwyn to
Ipswich
Kensington to
Keighley
Liverpool, Walton to
Lewes
Middlesbrough to
Meon Valley
Norwich North to
Newton Abbot
Orpington to
Ogmore
Poole to
Pendle
Reigate to
Redcar
St Ives to
Sunderland Central
Tynemouth to
Totnes
Worthing West to
Wansbeck
York Outer to
Yeovil