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

```
library(broom)
```

```
## 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))
```

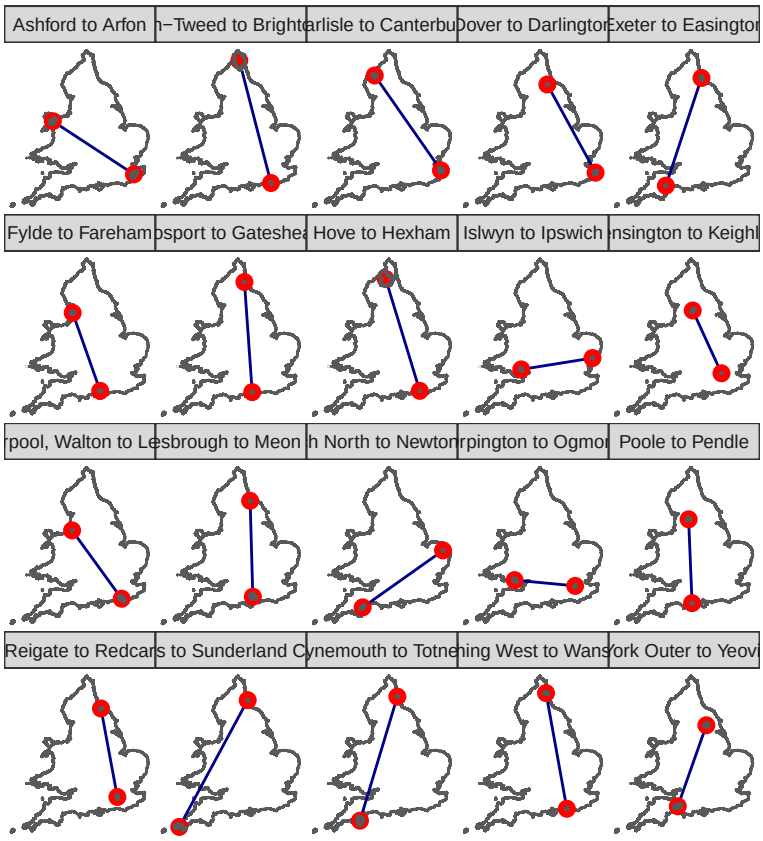
```
## # A tibble: 6 x 4
##   to                 distance from               first_letter
##   <chr>                 <dbl> <chr>              <chr>
## 1 St Ives             601117. Sunderland Central S
## 2 Tynemouth           540605. Totnes             T
## 3 Berwick-Upon-Tweed  528160. Brighton, Kemptown 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 <- "../../static/files/constituency_distances/england_wales_shape.rds" %>%
  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 ")) %>%
              select(first_letter, journey),
            by = "first_letter")

#get the center coordinates of constituencies
```

```r
#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 ")) %>%
              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)
```



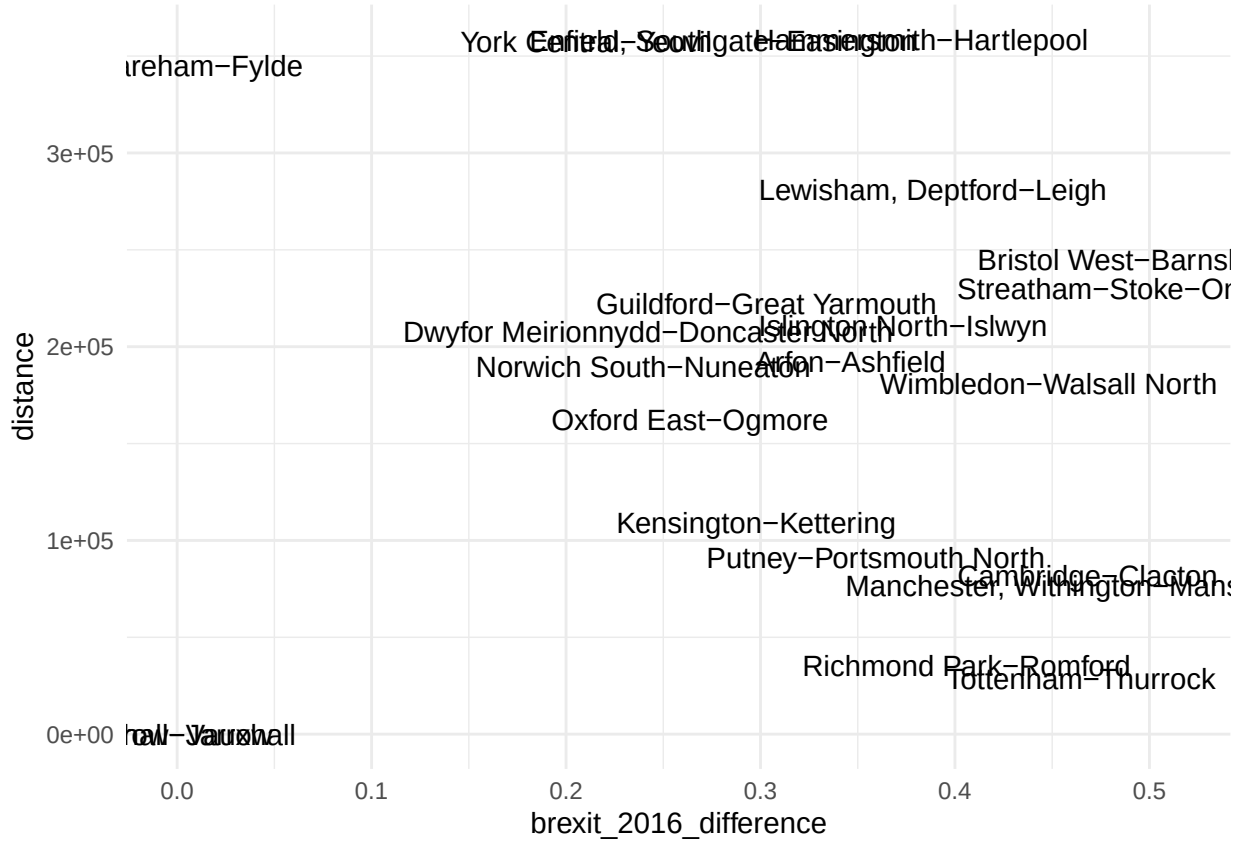

```
#load data on voting in the 2019 general election
#2016 brexit vote based on Hanretty work also included
votes_data <- readRDS("../../static/files/constituency_distances/ge2019_results.rds") %>%
  select(WSTnm = constituency_name, winner = first_party, votes = electorate,
         con, lab, ld, brexit, green, other, brexit_hanretty) %>%
  #convert to vote fractions
  modify_at(c("con", "lab", "ld", "brexit", "green", "other"), function(x) x/.$votes) %>%
  #take only relevant constituencies
  filter(WSTnm %in% constituency_geography$WSTnm)

head(votes_data)
```

```
##                 WSTnm winner votes       con       lab         ld
## 1            Aberavon    Lab 50750 0.1284335 0.3351330 0.02112315
## 2           Aberconwy    Con 44699 0.3285756 0.2830712 0.04073917
## 3           Aldershot    Con 72617 0.3853092 0.1553631 0.09529449
## 4 Aldridge-Brownhills    Con 60138 0.4631015 0.1332602 0.03942599
## 5        Amber Valley    Con 69976 0.4157997 0.1744884 0.04105693
## 6               Arfon     PC 42215 0.1048916 0.2452446 0.00000000
##       brexit       green       other brexit_hanretty
## 1 0.06124138 0.008866995 0.01440394       0.6012448
## 2 0.00000000 0.000000000 0.00000000       0.5219712
## 3 0.00000000 0.024099040 0.00000000       0.5789777
## 4 0.00000000 0.012820513 0.00558715       0.6779635
## 5 0.00000000 0.019835372 0.00000000       0.6529912
## 6 0.02745470 0.000000000 0.00000000       0.3584544
```

```r
#find the largest gap in 2016 brexit vote between constituencies
#which same first letter
brexit_differences <- votes_data %>%
  left_join(., constituency_names, by = "WSTnm") %>%
  split(f = .$first_letter) %>%
  map_df(., function(data) {
    difference <- outer(data$brexit_hanretty, data$brexit_hanretty, "-") %>%
      as.data.frame() %>%
      mutate(from = data$WSTnm)
    names(difference)[1:(ncol(difference) - 1)] <- data$WSTnm
    df <- difference %>%
      pivot_longer(cols = -starts_with("from"),
                   names_to = "to",
                   values_to = "brexit_2016_difference") %>%
      mutate(first_letter = unique(data$first_letter))
  }) %>%
  #arrange by greatest difference
  arrange(-brexit_2016_difference)

head(brexit_differences, n = 10)
```

```
## # A tibble: 10 x 4
##    from                 to           brexit_2016_difference first_letter
##    <chr>                <chr>                         <dbl> <chr>
##  1 Stoke-On-Trent North Streatham                     0.516 S
##  2 Stoke-On-Trent South Streatham                     0.506 S
##  3 Barnsley East        Bristol West                  0.503 B
##  4 Bolsover             Bristol West                  0.497 B
##  5 Barnsley East        Battersea                     0.489 B
##  6 Bolsover             Battersea                     0.483 B
##  7 Scunthorpe           Streatham                     0.481 S
##  8 Bassetlaw            Bristol West                  0.476 B
##  9 Barnsley Central     Bristol West                  0.475 B
## 10 Blackpool South      Bristol West                  0.471 B
```

```r
#plot the distances between constituencies of same first letter with
#greatest difference in 2016 brexit vote
brexit_distance_plot <- brexit_differences %>%
  filter(!duplicated(first_letter)) %>%
  left_join(constituency_interdistances, by = c("to", "from")) %>%
  mutate(journey = paste(to, from, sep = "-")) %>%
  select(journey, first_letter.x, brexit_2016_difference, distance) %>%
  ggplot(aes(x = brexit_2016_difference, y = distance, label = journey)) +
  geom_text() +
  theme_minimal()

brexit_distance_plot
```




```
#load the raw values from the census data for each output area
census_oa_data <- census_oa_data %>%
  #select only integer data (counts not percentages)
  select(OAid = GeographyCode, which(sapply(.,class)=="integer"))

#load the lookup between output areas to westminster constituency
oa_to_westminster <- readRDS("../../static/files/constituency_distances/oa_to_westminster.rds") %>%
  select(OAid = OA11CD, WSTid = PCON11CD, WSTnm = PCON11NM, WSTperc = OA11PERCENT) %>%
  #select only english constituencies
  filter(WSTid %in% constituency_names$WSTid)


#gather the census data by westminster constituency
census_data_westminster <- left_join(census_oa_data, oa_to_westminster, by = "OAid") %>%
  filter(!is.na(WSTid)) %>%
  #for output areas split between constituencies guesstimate the correct amounts
  mutate_if(is.integer, funs(round(. * (WSTperc/100)))) %>%
  select(-WSTnm, -WSTperc, -OAid) %>%
  #sum the counts per constituency for each statistic
  group_by(WSTid) %>%
  summarise_if(is.numeric, sum, na.rm = TRUE) %>%
  #turn into percentages from the total number of people (KS101)
  modify_if(is.numeric, function(x) x/.$KS101EW0001) %>%
  #arrange by name
  arrange(WSTid)
```

```r
#only preview the first few columns as we have ~400 total
head(census_data_westminster[1:8])
```

```
## # A tibble: 6 x 8
##   WSTid KS101EW0001 KS101EW0002 KS101EW0003 KS101EW0004 KS101EW0005
##   <chr>       <dbl>       <dbl>       <dbl>       <dbl>       <dbl>
## 1 E140~           1       0.499       0.501       0.980     0.0195
## 2 E140~           1       0.489       0.511       0.994     0.00574
## 3 E140~           1       0.492       0.508       0.991     0.00885
## 4 E140~           1       0.492       0.508       0.989     0.0111
## 5 E140~           1       0.485       0.515       0.992     0.00791
## 6 E140~           1       0.492       0.508       0.992     0.00766
## # ... with 2 more variables: KS101EW0006 <dbl>, KS102EW0001 <dbl>
```

```r
#load the terminology for each census statistic
census_index <- readRDS("../../static/files/constituency_distances/census_names.rds")

head(select(census_index, Code, Meaning))
```

```
##          Code
## 2 KS101EW0001
## 3 KS101EW0002
## 4 KS101EW0003
## 5 KS101EW0004
## 6 KS101EW0005
## 7 KS101EW0006
##                                                                                  Meaning
## 2                                                                  All categories: Sex
## 3                                                                                Males
## 4                                                                              Females
## 5                                                                 Lives in a household
## 6                                                  Lives in a communal establishment
## 7 Schoolchild or full-time student aged 4 and over at their non term-time address
```

```r
census_data_westminster <- census_data_westminster %>%
  #a few codes missing from the index
  .[-which(!names(.)[2:ncol(.)] %in% census_index$Code)] %>%
  .[c(1, which(!apply(.[2:ncol(.)],2,function(x) var(x,na.rm=T)==0))+1)]
```

```r
get_correlations_tidy <- function(demography, dependent_var) {
  #first split up the demography data by variable so we can independently
  #correlate each against the dependent variable
  split_demog <- demography %>%
    column_to_rownames("WSTid") %>%
    t() %>%
    split(f = rownames(.))

  #run the values for each variables against the dependent_var
  correlations <- map_df(split_demog, function(x) {
    regression <- lm(dependent_var ~ x)
    adj_r_squared <- summary(regression)$adj.r.squared
    f_stat <- summary(regression)$fstatistic[1]
```

```
    df <- summary(regression) %>%
      #tidy it to bind to df
      broom::tidy() %>%
      filter(term != "(Intercept)") %>%
      mutate(adj_r = adj_r_squared, f_stat)
  })

  tidy_df <- correlations %>%
    #left join in the meaning for each variable
    mutate(Code = names(demography[2:ncol(demography)])) %>%
    left_join(., select(census_index, Code, Meaning), by = "Code") %>%
    arrange(-abs(statistic)) %>%
    select(-term)

  #return this data frame
  return(tidy_df)
}
```

```
lr_margin <- votes_data %>%
  left_join(constituency_names, by = "WSTnm") %>%
  #must line up in order with census data
  arrange(WSTid) %>%
  #assuming a simple left vs right decision for voters
  mutate(left = lab + ld + green,
         right = con + brexit) %>%
  #take the difference between left and right sum votes for each constituency
  mutate(margin = left - right) %>%
  .$margin

#run in the above function
lr_correlations <- get_correlations_tidy(census_data_westminster, lr_margin)

head(select(lr_correlations, Code, Meaning, statistic, p.value))
```

```
## # A tibble: 6 x 4
##   Code      Meaning                                          statistic  p.value
##   <chr>     <fct>                                                <dbl>    <dbl>
## 1 KS104EW0~ Not living in a couple: Single (never marri~      26.0 1.86e-83
## 2 KS404EW0~ All categories: Car or van availability          -23.4 1.67e-73
## 3 KS104EW0~ Living in a couple: Married or in a registe~     -23.2 1.44e-72
## 4 KS105EW0~ One family only: Married or same-sex civil ~     -22.2 1.78e-68
## 5 KS103EW0~ Married                                          -22.0 4.93e-68
## 6 KS404EW0~ 2 cars or vans in household                      -22.0 5.77e-68
```

```
lr_correlations <- lr_correlations %>%
  #lots of these stats are self-correlated
  #e.g. 3 cars in household vs 4+ cars in household
  mutate(stat_category = gsub("(.{5})(.*)", "\\1", Code)) %>%
  group_by(stat_category) %>%
  #take only the strongest correlated variable from each 'category'
  mutate(duplicate_n = 1:n()) %>%
  ungroup() %>%
  filter(duplicate_n == 1 & abs(statistic) > 10 & !duplicated(Meaning))
```

```
right_variables <- lr_correlations %>%
  mutate(census_info = paste(Code, Meaning)) %>%
  filter(statistic < 0) %>%
  .$census_info

left_variables <- lr_correlations %>%
  mutate(census_info = paste(Code, Meaning)) %>%
  filter(statistic > 0) %>%
  .$census_info

right_variables
```

```
##  [1] "KS404EW0007 All categories: Car or van availability"
##  [2] "KS105EW0005 One family only: Married or same-sex civil partnership couple: No children"
##  [3] "KS103EW0003 Married"
##  [4] "KS501EW0004 Highest level of qualification: Level 2 qualifications"
##  [5] "KS102EW0013 Age 60 to 64"
##  [6] "KS401EW0008 Whole house or bungalow: Detached"
##  [7] "KS603EW0002 Economically active: Employee: Part-time"
##  [8] "KS609EW0006 5. Skilled trades occupations"
##  [9] "KS605EW0007 F Construction"
## [10] "KS402EW0002 Owned: Owned outright"
## [11] "KS301EW0014 Provides 1 to 19 hours unpaid care a week"
## [12] "KS209EW0002 Christian"
```

```
left_variables
```

```
##  [1] "KS104EW0004 Not living in a couple: Single (never married or never registered a same-sex civil
##  [2] "KS403EW0004 Occupancy rating (rooms) of -1 or less"
##  [3] "KS201EW0007 Mixed/multiple ethnic group: White and Black African"
##  [4] "KS604EW0008 Males: Part-time: 16 to 30 hours worked"
##  [5] "KS612EW0012 L14.1 Never worked"
##  [6] "KS613EW0014 Not classified"
##  [7] "KS202EW0039 Other identities only"
##  [8] "KS206EW0005 No people in household have English as a main language (English or Welsh in Wales)
##  [9] "KS204EW0010 Other countries"
## [10] "KS205EW0008 Middle East and Asia"
## [11] "KS107EW0012 Female lone parent: Not in employment"
## [12] "KS106EW0002 No adults in employment in household: With dependent children"
```

```
brexit_vote <- votes_data %>%
  left_join(constituency_names, by = "WSTnm") %>%
  #must line up in order with census data
  arrange(WSTid) %>%
  .$brexit_hanretty

brexit_correlations <- get_correlations_tidy(census_data_westminster, brexit_vote) %>%
  #lots of these stats are self-correlated
  #e.g. 3 cars in household vs 4+ cars in household
  mutate(stat_category = gsub("(.{5})(.*)", "\\1", Code)) %>%
  group_by(stat_category) %>%
  #take only the strongest correlated variable from each 'category'
```

```
  mutate(duplicate_n = 1:n()) %>%
  ungroup() %>%
  filter(duplicate_n == 1 & abs(statistic) > 10 & !duplicated(Meaning))
```

```
## Warning: Column `Code` joining character vector and factor, coercing into
## character vector
```

```
head(brexit_correlations)
```

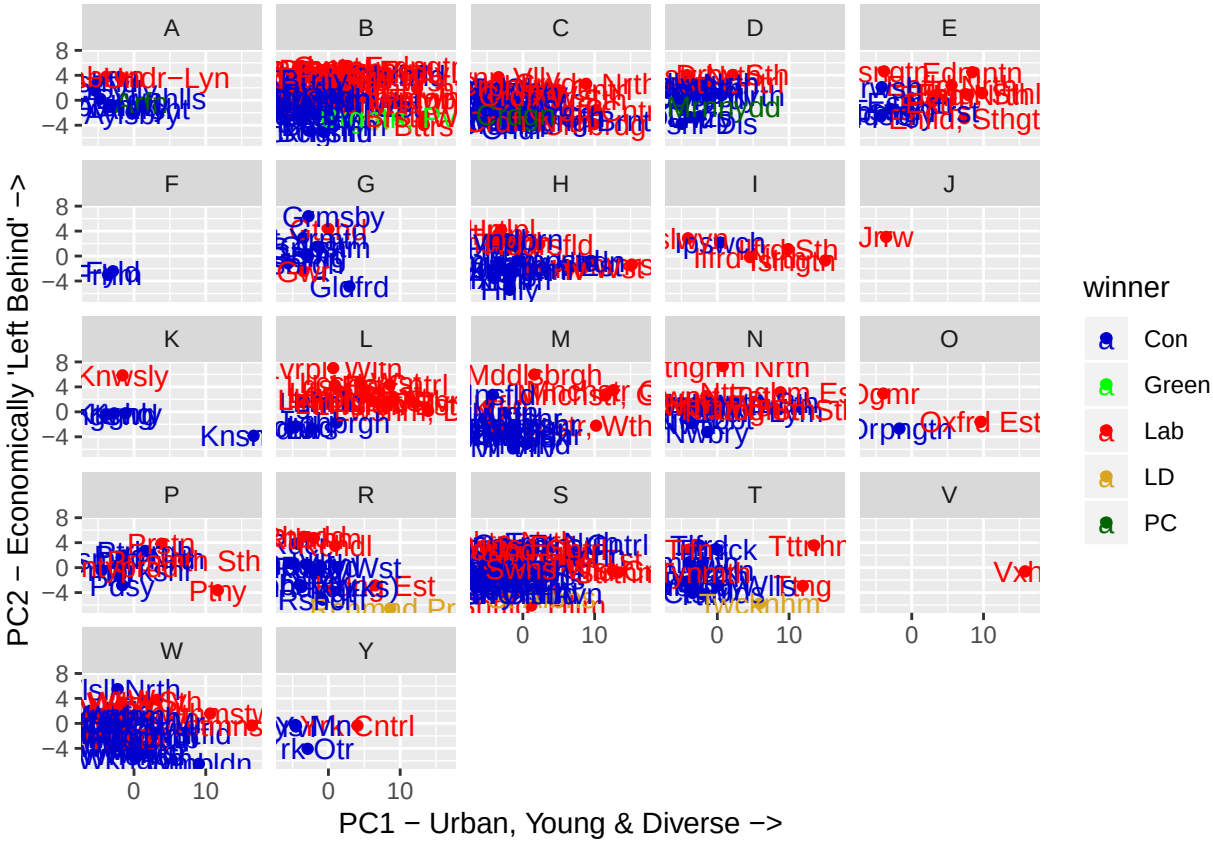
```
## # A tibble: 6 x 10
##   estimate std.error statistic   p.value adj_r f_stat Code  Meaning
##      <dbl>     <dbl>     <dbl>     <dbl> <dbl>  <dbl> <chr> <fct>
## 1     5.27     0.171      30.8 2.32e-101 0.731   948. KS50~ Highes~
## 2     6.08     0.239      25.4 3.36e- 81 0.648   645. KS61~ 6. Sem~
## 3    -3.14     0.126     -25.0 1.52e- 79 0.641   623. KS60~ 2. Pro~
## 4     9.04     0.380      23.8 5.00e- 75 0.618   567. KS60~ 8. Pro~
## 5     7.65     0.329      23.3 6.44e- 73 0.608   542. KS61~ 5. Low~
## 6    -2.20     0.103     -21.3 5.47e- 65 0.564   453. KS30~ Very g~
## # ... with 2 more variables: stat_category <chr>, duplicate_n <int>
```

```
lr_correlations$Meaning[which(lr_correlations$Code %in% brexit_correlations$Code)]
```

```
## [1] Not living in a couple: Single (never married or never registered a same-sex civil partnership)
## [2] Economically active: Employee: Part-time
## [3] Occupancy rating (rooms) of -1 or less
## 793 Levels: 0 or multiple adults in household ... Widowed or surviving partner from a same-sex civil
```

```
pca_census <- census_data_westminster %>%
  #take only the variable that strongly correlate with 2019/brexit vote
  select(unique(c(lr_correlations$Code, brexit_correlations$Code))) %>%
  #scale before pca
  scale()

#run the pca
#take first 3 components
demographic_pca <- prcomp(pca_census)$x %>%
  as.data.frame() %>%
  .[1:3] %>%
  #add back in ID column
  mutate(WSTid = census_data_westminster$WSTid) %>%
  #join in additional dta for plotting
  left_join(., constituency_names, by = "WSTid") %>%
  left_join(., select(votes_data, WSTnm, winner), by = "WSTnm")

#plot
demographic_pca_plot <- demographic_pca %>%
  ggplot(aes(x = PC1, y = PC2, label = gsub("a|e|i|o|u", "", WSTnm), colour = winner)) +
  geom_point() +
  geom_text() +
  scale_colour_manual(values = c("mediumblue", "green", "red", "goldenrod", "darkgreen")) +
  labs(x = "PC1 - Urban, Young & Diverse ->",
```

```
       y = "PC2 - Economically 'Left Behind' ->") +
  facet_wrap(~first_letter)

demographic_pca_plot
```




```
distances <- demographic_pca %>%
  split(f = .$first_letter) %>%
  map_df(., function(data) {
    distances <- (outer(data$PC1, data$PC1, "-")^2 + outer(data$PC2, data$PC2, "-")^2) %>%
    sqrt() %>%
    as.data.frame() %>%
    mutate(from = data$WSTnm)
    names(distances)[1:(ncol(distances)-1)] <- as.character(data$WSTnm)
    df <- pivot_longer(distances, -starts_with("from"),
                       names_to = "to",
                       values_to = "pca_distance") %>%
      mutate(pca_distance = abs(pca_distance))
    return(df)
  }) %>%
  filter(!duplicated(pca_distance) & pca_distance != 0)
```

```
library(ggrepel)
all_distances <- distances %>%
  left_join(brexit_differences, by = c("from", "to")) %>%
  mutate(brexit_2016_difference = abs(brexit_2016_difference)) %>%
```

```
  select(-first_letter) %>%
  left_join(constituency_interdistances, by = c("from", "to")) %>%
  mutate(label = case_when(
    distance > 400000 & pca_distance > 15 & brexit_2016_difference > 0.3 ~ paste(to, from, sep = "-")
  )) %>%
  mutate(distance = distance / 1000)

p <- ggplot(all_distances, aes(x = distance, y = pca_distance, size = brexit_2016_difference)) +
  geom_point(alpha = 0.2) +
  geom_point(data = filter(all_distances, !is.na(label))) +
  geom_text_repel(aes(label = label)) +
  scale_size_continuous(name = "diff Brexit\n 2016 vote", range = c(0.5, 5)) +
  labs(x = "Geographic Distances between Constituences (/km",
       y = "'Distance' between Constituencies Demograph (2011 Census)") +
  theme_minimal()

plot(p)
```

```
## Warning: Removed 4608 rows containing missing values (geom_text_repel).
```