API & AI Practical

Laura Graham

2023-03-06

Introduction

In this practical we are going to be examining what people enjoy about Sutton Park in Birmingham, and how these are spatially distributed. Below is a map of Sutton Park with details of both environmental and cultural features.

The data source we will use is photographs, and photograph metadata, from the image sharing website Flickr. This data source has been used in research to understand spatial distribution and drivers of cultural recreation, including visitation rates to parks, and what people focus on while they are there.

In this practical we will:

1. Download metadata about photos from Flickr for Sutton Park in the year 2020
2. Use Natural Language Processing to convert text to a machine understandable 'token'
3. Use cluster analysis to establish 'patterns' in the photo metadata, and ultimately assign 'categories' of photographs
4. Use sentiment analysis on the textual metadata to gain further information about people's feelings during their park visit
5. Visualise our results

All code that is in grey boxes should be copied into your R script and run

To start with, run this line to install all the required packages.

```
source("install_packages.R")
```

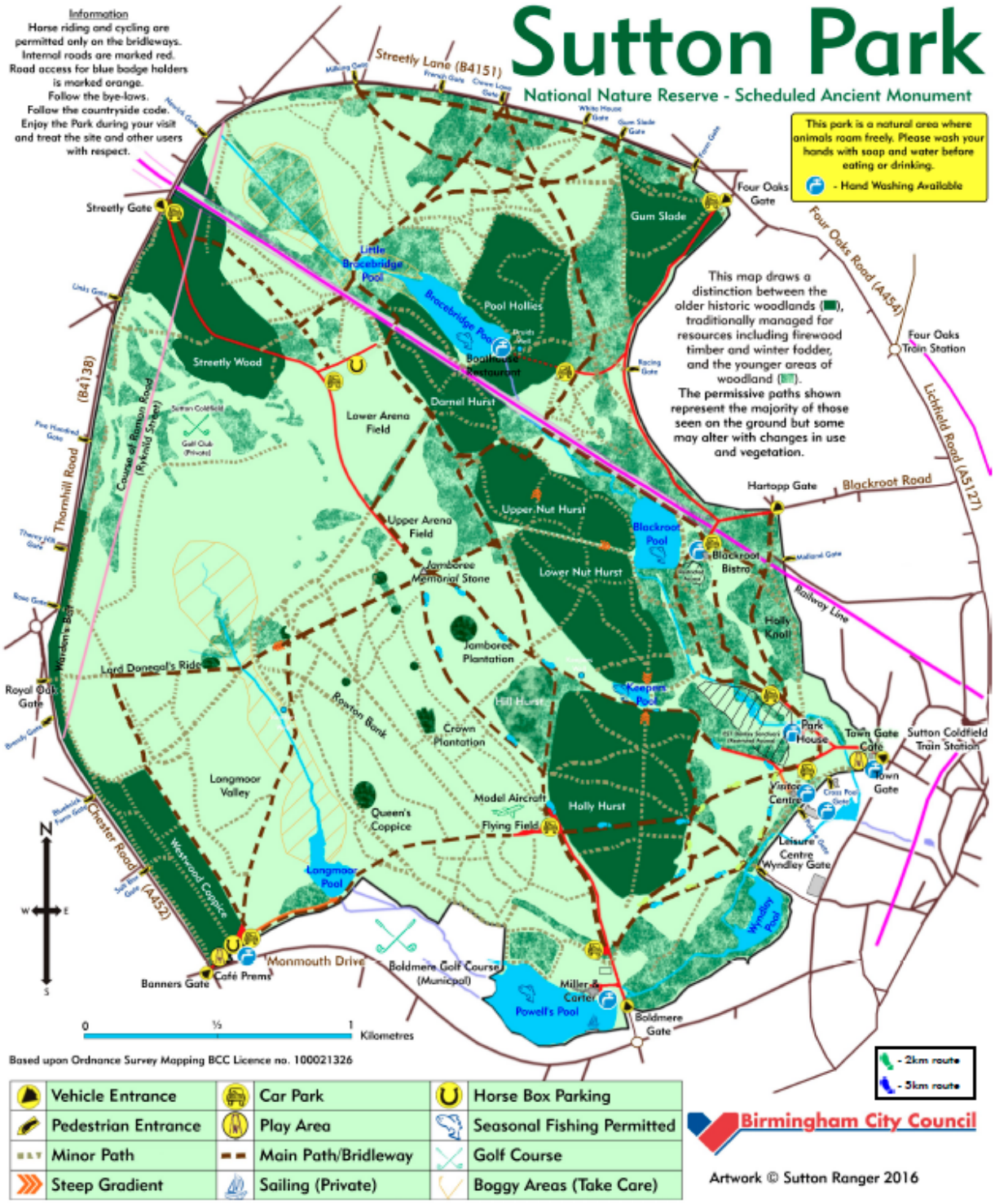
A note about the `tidyverse` R packages

I have used a collection of R packages known as the `tidyverse`.

Make sure to run this to load the package

```
library(tidyverse, quietly = TRUE)
```

If you are not already familiar with it, something that can be confusing is the `%>%` operator - known as a pipe. This essentially pushes the result from the bit before the pipe into the first argument of the bit after the pipe. It's not super important, but will help you understand the code structure.



Download photograph metadata

For this we will use the R package `photosearcher`. This accesses the Flickr API and returns metadata and photographs based on a user search.

It is possible to search using a keyword, location, and/or range of dates. Here we will only specify the location and a range of dates.

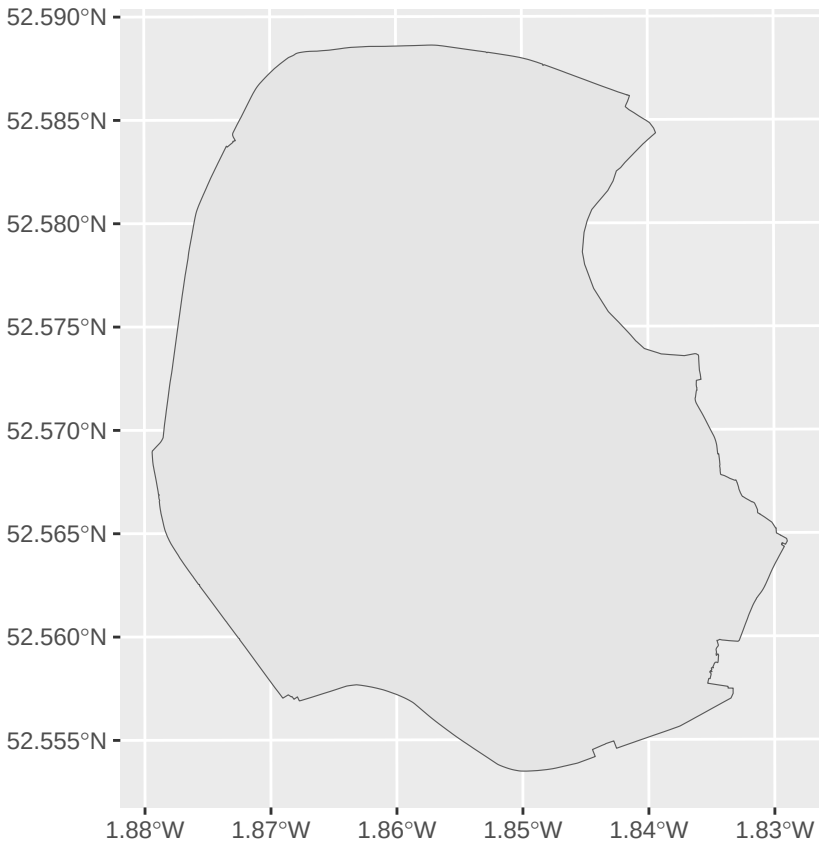
First, we will need to load a boundary file for Sutton Park. We will do so using a package called `sf`. I won't go into too much detail on this, but essentially it allows you to do GIS in R. Click on the link for more details.

```
library(sf)

boundary <- st_read("Sutton Park Boundary_region.shp")
```

```
## Reading layer 'Sutton Park Boundary_region' from data source
##   'C:\Users\grahamlz\OneDrive - University of Birmingham\TEACHING\Digital Data Capture\Practical\Sut
##   using driver 'ESRI Shapefile'
## Simple feature collection with 1 feature and 1 field
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 408271.9 ymin: 295081.1 xmax: 411691 ymax: 298989.2
## Projected CRS: OSGB36 / British National Grid
```

We can plot this data using `ggplot`:

```
library(ggplot2)

ggplot(boundary) +
  geom_sf()
```



Now we can use the function `photo_search` to get the metadata for the photos we need:

```r
library(photosearcher)

sutton_photos <- photo_search(
  mindate_taken = "2020-01-01",
  maxdate_taken = "2026-01-01",
  sf_layer = boundary,
  has_geo = TRUE
)
```

Explore the data with the below commands:

```r
str(sutton_photos)
```

Keyword analysis from photo data

Here we will prepare the textual metadata for analysis.

```r
# Combine title, description, tags into one text column per photo
library(textclean)
sutton_text <- sutton_photos %>%
  mutate(
    text = paste(title, description, tags),
    text = textclean::replace_url(text),
    text = stringr::str_squish(text)
  ) %>%
  filter(!is.na(text), text != "")
```

Tokenise step (i.e. make it machine readable)

```r
library(tidytext)
data("stop_words")

tidy_tokens <- sutton_text %>%
  select(url_l, longitude, latitude, title, description, tags, text) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by = "word") %>%
  filter(str_detect(word, "^[a-z][a-z'-]*$")) %>%   # keep simple words
  count(url_l, word, sort = FALSE)                  # term frequency per photo
```

Apply cluster analysis to photo text descriptions

Next we will apply cluster analysis to the photo text to group them. Cluster analysis aims to find patterns in the data by grouping similar data points. In this case, the groupings will be formed based on how often pairs of keywords appear together.

The next bit of code creates a photo-by-keyword matrix, where only keywords appearing in 5 or more photos are selected:

```r
# Create a photo x word matrix which shows 1 where a word is present in the photo text, 0 when it is no
photo_word <- tidy_tokens %>%
  mutate(n = 1) %>%
  pivot_wider(names_from = word, values_from = n, values_fill = 0)

# Keep only words that appear in >=5 photos
word_counts <- tidy_tokens %>% count(word, name = "photo_count")
```

```
keep_words <- word_counts %>% filter(photo_count >= 5) %>% pull(word)

photo_word_mat <- tidy_tokens %>%
  filter(word %in% keep_words) %>%
  mutate(n = 1) %>%
  pivot_wider(names_from = word, values_from = n, values_fill = 0) %>%
  select(-url_l) %>%
  as.matrix()
```

Next we convert this to a keyword-by-keyword matrix using some matrix algebra. The numbers in the matrix represent how many times each pair of keywords appears together

```
keyword_cooccurrance <- t(photo_word_mat) %*% photo_word_mat
```

We can look at the first 5 pairs to get an idea of what the matrix looks like

```
keyword_cooccurrance[1:5, 1:5]
```

```
##            amp bird birds birmingham canon
## amp        143  117   115         12    16
## bird       117  917   153         30    13
## birds      115  153   156         24    12
## birmingham  12   30    24        129     3
## canon       16   13    12          3    18
```

We use the `igraph` R package to do the cluster analysis. First, the matrix is converted to a graph, then we use a Walktrap algorithm (as in the Lee et al. 2018 paper) to define the clusters:

```
library(igraph, warn.conflicts = FALSE)

graph <- graph.adjacency(keyword_cooccurrance,
                         weighted=TRUE,
                         mode="undirected",
                         diag=FALSE)

clusters <- cluster_walktrap(graph)
```

This next bit applies an analysis which finds out which keywords are most important in defining the cluster:

```
keyword_importance <- map_df(1:max(clusters$membership), function(i) {
  graph_subset <- keyword_cooccurrance[clusters$membership == i,clusters$membership == i]
  graph_subset <- graph.adjacency(graph_subset,
                                  weighted=TRUE,
                                  mode="undirected",
                                  diag=FALSE)
  eigens <- eigen_centrality(graph_subset)$vector
  eigens_df <- tibble(eigenvalue = eigens, keyword = names(eigens), cluster = i) %>%
    arrange(-eigenvalue)

})
```

We will use the top 10 keywords to assign a meaning to our clusters.

```
keyword_importance %>% group_by(cluster) %>%
  slice(1:10)
```

```
## # A tibble: 76 x 3
## # Groups:   cluster [9]
```

```
##    eigenvalue keyword      cluster
##         <dbl> <chr>          <int>
##  1      1     wild               1
##  2      0.992 national           1
##  3      0.992 reserve            1
##  4      0.989 water              1
##  5      0.987 lake               1
##  6      0.986 wildlife           1
##  7      0.976 photography        1
##  8      0.976 geographic         1
##  9      0.976 natural            1
## 10      0.976 photographer       1
## # i 66 more rows
```

- Look at the top 10 keywords for each cluster and think about what it is describing.
- Come up with names for each cluster

Now we can assign the most representative cluster for each photograph:

```
word_to_cluster <- tibble(keyword = V(graph)$name, cluster = clusters$membership)


assign_cluster <- tidy_tokens %>%
  filter(word %in% word_to_cluster$keyword) %>%
  inner_join(word_to_cluster, by = c("word" = "keyword")) %>%
  count(url_l, cluster, name = "votes") %>%
  group_by(url_l) %>%
  slice_max(votes, n = 1, with_ties = TRUE) %>%
  slice(1) %>%                              # break ties deterministically
  ungroup() %>%
  mutate(cluster = factor(cluster, labels = paste("Cluster", sort(unique(cluster)))))
```

In this next bit, you will add the names you have given to your clusters - make sure to replace the text which says "cluster x" with your name.

```
assign_cluster$cluster <- factor(assign_cluster$cluster,
                                 labels = c("Cluster 1",
                                            "Cluster 2",
                                            "Cluster 3",
                                            "Cluster 4",
                                            .....))
```

Visualise clustering results

We will add the photograph locations onto the map and colour the points by the cluster

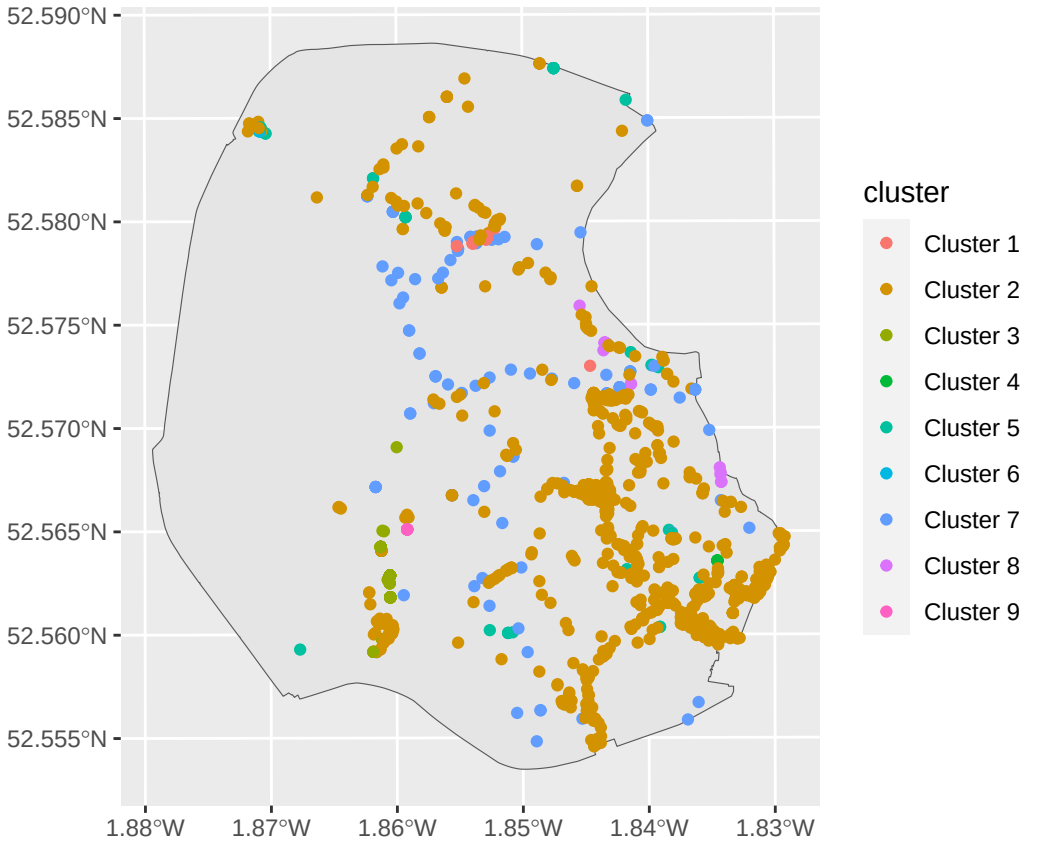
The first bit of code allows us to take a data frame with x, y locations and make it into a vector shapefile type object:

```
sutton_photo_clusters <- sutton_photos %>%
  inner_join(assign_cluster) %>%
  st_as_sf(coords = c("longitude", "latitude"), crs = st_crs(4326))
```

```
## Joining with `by = join_by(url_l)`
```

And now we can plot the results:

```
ggplot() +
  geom_sf(data = boundary) +
  geom_sf(data = sutton_photo_clusters, aes(colour = cluster))
```



Have a look at the spatial locations of the clusters and compare with the Sutton Park site map. Do these make sense? Discuss with your neighbour.

Apply sentiment analysis to textual metadata

Finally we will apply sentiment analysis to the textual metadata provided in the `title`, `description` and `tags` fields.

First we need to get these fields into one column:

```
# col with all text data
sutton_photo_clusters$text <- paste2(sutton_photo_clusters$title,
                                     sutton_photo_clusters$description,
                                     sutton_photo_clusters$tags)
```

Next we use the `tidytext` R package to do the analysis - this uses an extensive dictionary called `afinn` which assigns a sentiment value to a word.

```
# unnest words - one row per word
sutton_photo_text <- sutton_photo_clusters %>%
  unnest_tokens(word, text)

sentiment <- sutton_photo_text %>%
  inner_join(get_sentiments("afinn")) %>%
```

```
  group_by(url_l) %>%
  summarise(sentiment = sum(value)) %>%
  st_set_geometry(NULL) %>%
  inner_join(sutton_photo_clusters) %>%
  arrange(-sentiment) %>%
  mutate(id = 1:n())
```

```
## Joining with `by = join_by(word)`
## Joining with `by = join_by(url_l)`
```

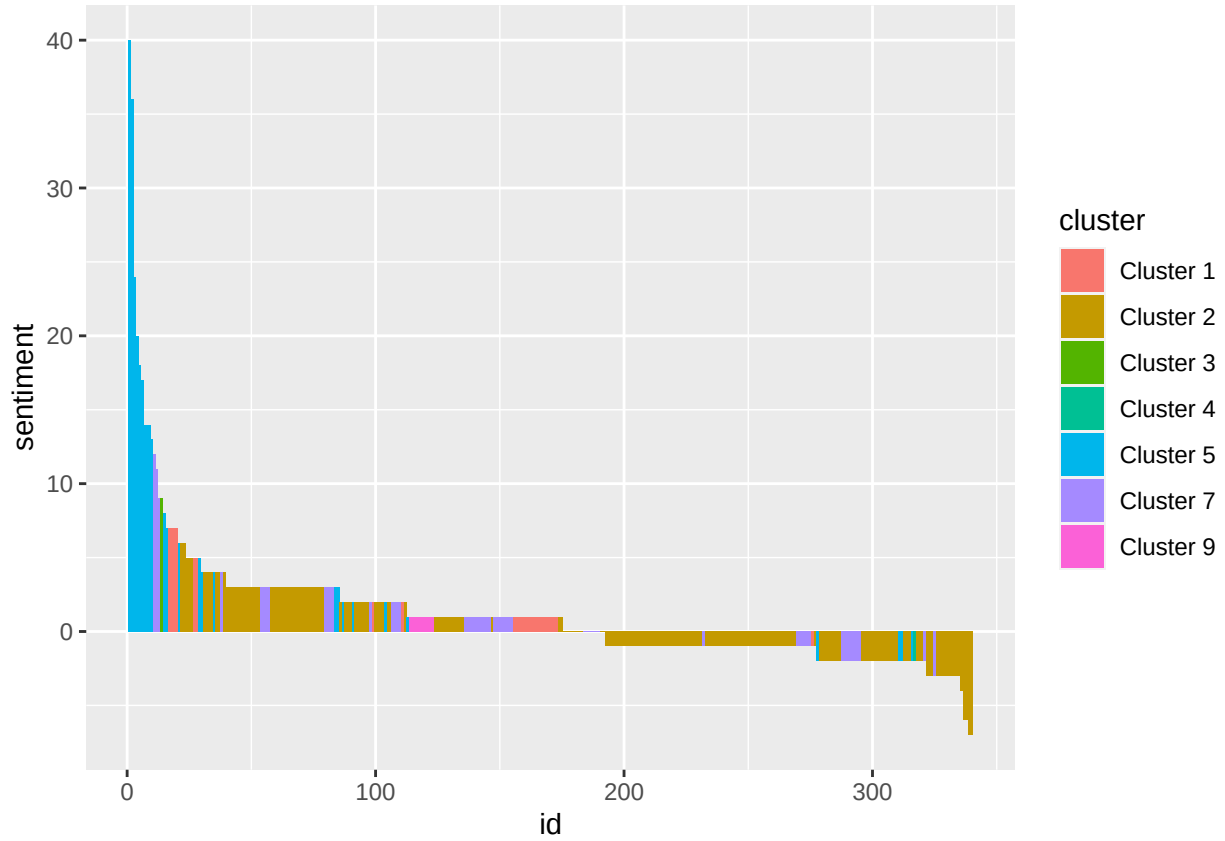
Visualise results of sentiment analysis

Take a look at the text column and the sentiment column of some of the entries to see if it makes sense.

```
select(sentiment, text, sentiment)
```

Let's view the results by cluster:

```
ggplot(sentiment, aes(x = id, y = sentiment,
                      fill = cluster)) +
  geom_bar(stat = "identity")
```



- Are some clusters more negative, and some more positive?
- Do you agree with the results?