CW2_GY7708

219031729

4/18/2022

Course Work 2 GY7708

Geographic Information retrieval and Sentiment analysis

```
#Loading Libraries
library(sf)
library(tidyverse)
library(cowplot)
library(stringi)
library(tidytext)
library(wordcloud)
library(reshape2)
library(spdep)
library(tidylo)
library(topicmodels)
library(spdep)
```

Part 1

Loading Data

```
#Loading in the CSV file
wiki_geo <- read.csv('data/wikipedia_geotags_in_UK.csv')

#Loading boundary shapefile of the study area
hackneyshp <- st_read('data/hackney/Export_Output.shp')
```

```
## Reading layer `Export_Output' from data source
##   `/home/kal41/GY7708_S2/Postgresql2/data/hackney/Export_Output.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 1 feature and 7 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 531479.8 ymin: 181840.5 xmax: 537640.8 ymax: 188327.4
## Projected CRS: OSGB 1936 / British National Grid
```

```
## The geometry type is Polygon and the CRS is OSGB 1936 / British National Grid
```

Plotting Boundary Shapefile

```
#Ploting the boundary shapefile of the study area
plot(hackneyshp$geometry)
```

Data Preprocessing

```
#Filtering the CSV to my allocated area
wiki_geo <- wiki_geo %>%
  filter(LAD21NM == 'Hackney') %>% #selecting my allocated LAD
  filter(gt_primary == 1) #Removing pages without a geotag

#Selecting only columns that will be merged with the data
wiki_geo_coord <- wiki_geo %>% select(gt_id, gt_lat, gt_lon, page_title)
```

Text Mining/Web Scraping

```
#Building function for the extraction of Wikipedia pages
#in my allocated LAD.
unest_function <- function(page_title) {
a_page_summary <-
  httr::GET(
    # Base API URL
    url = "https://en.wikipedia.org/w/api.php",
    # API query definition
    query = list(
      # Use JSON data format
      format = "json",
      action = "query",
      # Only retrieve the intro
      prop = "extracts",
      exintro = 1,
      explaintext = 1,
      redirects = 1,
```

```
      # Set the title
      titles = page_title
    )
  ) %>%
  httr::content(
    as = "text",
    encoding = "UTF-8"
  ) %>%
  jsonlite::fromJSON() %>%
  # Extract the summary from the list
  magrittr::extract2("query") %>%
  magrittr::extract2("pages") %>%
  magrittr::extract2(1) %>%
  magrittr::extract2("extract")

#Converting the text to dataframe
a_page_summary <- as.data.frame(a_page_summary)


#creating a column to store each page title
a_page_summary <- a_page_summary %>%
  mutate(page_title)
return(a_page_summary)

}

# Creating an empty dataframe that will be used to house the data
page_word <- data.frame(
  page_title = character(),
  a_page_summary = character()
)


# Created a loop to run the above function for each of the pages in my allocated LAD.
for (i in 1:nrow(wiki_geo)) {
  page_title <- wiki_geo$page_title[i]
  page_word <- page_word %>%
    add_row(unnest_function(page_title)) #adding results from new pages
}
```

The number of Hackney pages extracted from Wikipedia is one lesser that the number in the #excel table (241:242). This is because the Wikipedia page for page_title The_Centre_of_Attention' is null; hence no data for The_Centre_of_Attention page.

Adding Spatial information to the data

```
#Adding the coordinate of the pages from the excel file
page_wordwtCd <- page_word %>% left_join(wiki_geo_coord, by = 'page_title')

#Converting the CRS of the data to British National Grid
page_word_Brt <- page_wordwtCd %>%
  st_as_sf(coords = c("gt_lon", "gt_lat"), crs = 4326) %>%
  st_transform(27700)
```

Tokenization of Text

```
#Tokenizing the text and transforming it into
#tidy data structure
page_word_Brtun <- page_word_Brt %>%
  unnest_tokens(word, a_page_summary) %>% #Creating token word from the sentences
  anti_join(get_stopwords()) #removing stopwords
```

Tokenization allows easy vectorization of the text data as vectoring is an important process when analyzing text with machine learning models.

Part 2: Spatial Frequency Analysis

- **Ordinary Word Frequency Spatial Analysis**: This is all about the variation of word count. The word count was done in the following ways:

1. Words frequency usage in the entire Hackney.

2. The total number of words used on each Hackney page.

- Word per page frequency. This shows the frequency of each word on each page.
  This helps determine the variation of word usage across all the pages in Hackney.
- **Term Frequency Analysis**: This is the measure of word frequency rate per the overall word count in the whole Hackney. It is often used to measure how important a word is.
- **Spatial Autocorrelation**: This is a correlation analysis that measures the randomness of a variable based on the value of its surrounding neighbours. The z-value from this statistic is used to determine a clustered, dispersed, or random spatial relationship.
- **TF-IDF Frequency Analysis**: it is known as term frequency-inverse document frequency. TF-IDF is an advancement of the 'Term Frequency' analysis. It is regarded as a better measure of word importance. Unlike term frequency, TF- does not just measure the relevance of a word in the document but calculates the importance of the word to the document among the subgroups in the document. It is calculated as the product of a word frequency and its inverse frequency (rarity) across the subgroups within the document. Higher TF-IDF value indicates higher relevance and vice-a-vice.
- **Weight Log odds**: This another measure of word usage across a collection of documents in a document. It uses the empirical Bayes approach to estimate and bind the posterior log odds ratios of a word. Although Weight Log odds works in a fairly similar manner as the TF-IDF, it is an advancement of the latter. The edge Weight Log odds has over TF-IDF is that it does not assign a zero value to a word used in all collection of the document. It lets us understand the relevance of every word even if common to all subgroups. For example, in the case of Hackney, the words "London" and "Hackney" is used in almost all the subgroups (page title), hence, TF-IDF is not able to capture the relevance of this words.

```
#Extracting the frequency of words
total_word_count <- page_word_Brtun %>%
  count(word,  sort = TRUE)
class(page_word_Brtun)
```

```
## [1] "sf"         "data.frame"
```

```
#Top 10 most used words in all the pages in Hackney
total_word_count %>%
  slice_max(n, n = 10) %>%
  knitr::kable(caption = "Top 10 Most Used Words in all Hackney Pages")
```

Table 1: Top 10 Most Used Words in all Hackney Pages

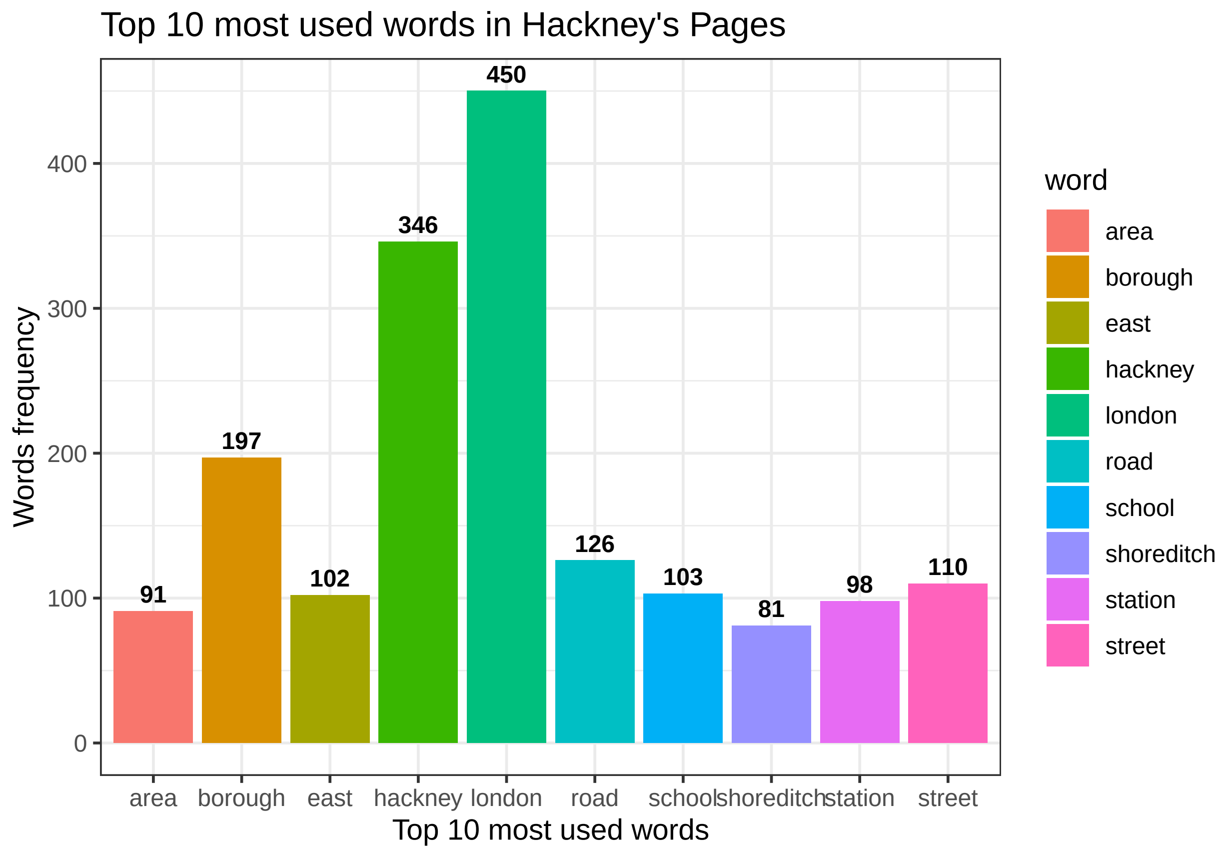
word	n	geometry
london	450	MULTIPOINT ((531726 186555....
hackney	346	MULTIPOINT ((531999.5 18684...
borough	197	MULTIPOINT ((531999.5 18684...
road	126	MULTIPOINT ((531726 186555....
street	110	MULTIPOINT ((532238.4 18299...
school	103	MULTIPOINT ((532448.5 18750...
east	102	MULTIPOINT ((532304.8 18310...
station	98	MULTIPOINT ((532045.1 18748...
area	91	MULTIPOINT ((532183.2 18321...
shoreditch	81	MULTIPOINT ((532464.1 18276...

```
total_word_count %>%
  slice_min(n, n = 10) %>%
  head(10) %>%
  knitr::kable(caption = "Bottom 10 Least Used Words in all Hackney Pages")
```

Table 2: Bottom 10 Least Used Words in all Hackney Pages

word	n	geometry
0.830	1	POINT (535500.3 185499.5)
07	1	POINT (533389.7 183026.6)
1,000	1	POINT (532847.6 182522.8)
1,500	1	POINT (534725.2 185382.4)
1.2	1	POINT (534546.5 184444.1)
1.6	1	POINT (533119.4 182588.9)
10,165	1	POINT (534363.2 184009.3)
10,290	1	POINT (534547.6 184904.4)
10,600	1	POINT (532847.6 182522.8)
10.9	1	POINT (535500.3 185499.5)

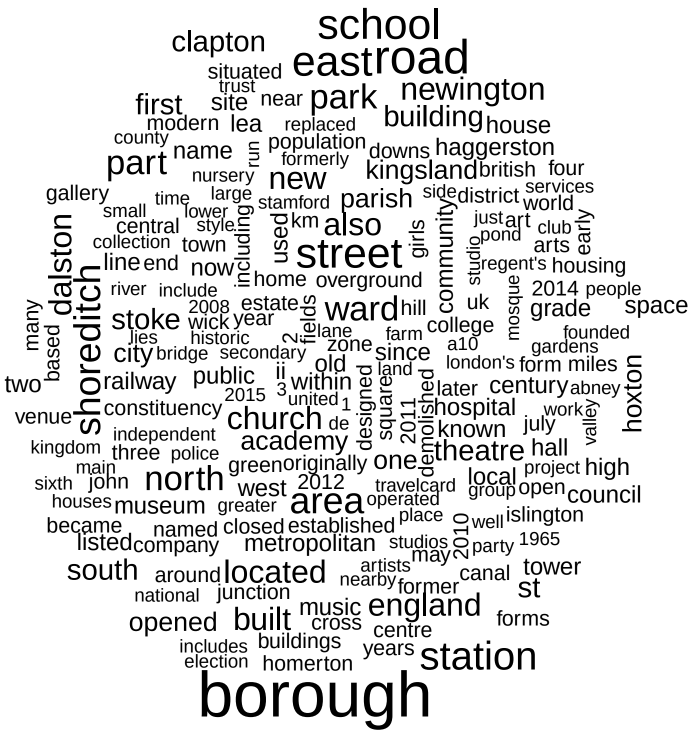
```
total_word_count %>%
  slice_max(n, n = 10) %>%
  ggplot(aes(word, n, fill = word)) +
  geom_col() +
  geom_text(aes(label = n), size = 3, fontface = "bold", vjust = -0.5) +
  labs(title = "Top 10 most used words in Hackney's Pages",
       x = "Top 10 most used words", y = "Words frequency") +
  theme_bw()
```



The most used non-stop words in my allocated area(Hackney) is London The most used word in all the Hackney pages are: "london, hackney, borough, road, street, school, east, station, area, shoredicth" This is becuase teh area is in east london. Also, it can be assumed that there are lots of schools and stations in the area. Meanwhile, the word "shoreditch" which is the 10th most used word is an important word in Hackney; it represents an administrative that consists of different important boundaries.

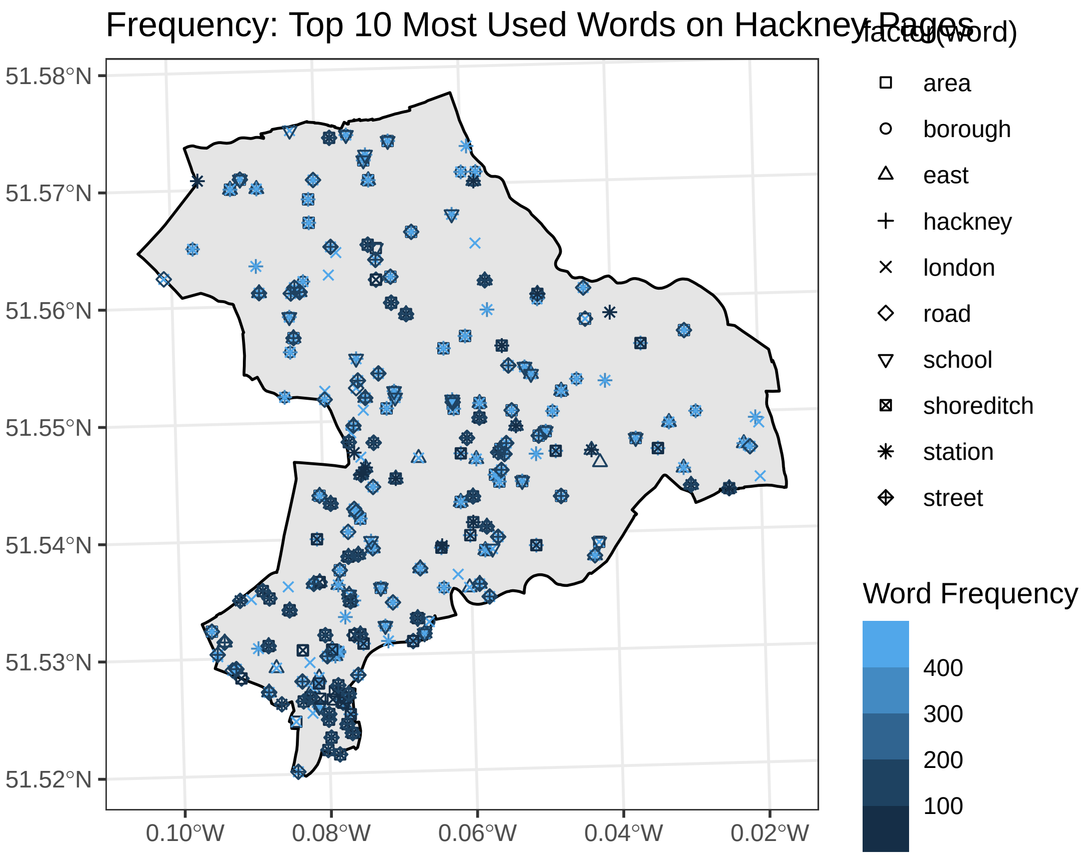
Word Cloud Representing

```
#Word Cloud representing each word with size based on their frequency
total_word_count %>%
  with(wordcloud(word, n, max.words = 200))
```



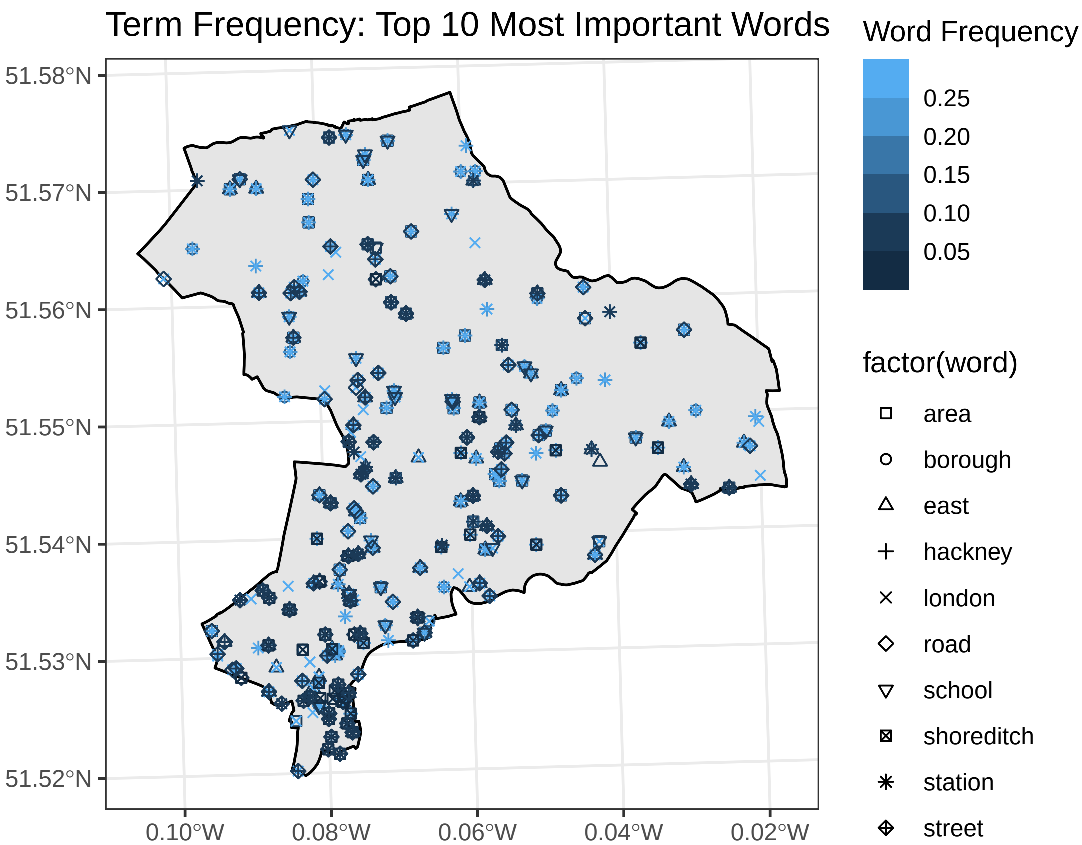
Word Usage Frequency Analysis

```r
#Top 10 most used words in all Hackney Pages
top_10_hackney_wds <- total_word_count %>%
  slice_max(n, n = 10)


#Map of Top 10 most used words in Hackney in Hackney
ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = top_10_hackney_wds, aes(color = n, shape = factor(word))) +
  scale_shape(name = "Words")+
  scale_color_steps(name = 'Word Frequency')+
  theme_bw()+
  labs(title = 'Frequency: Top 10 Most Used Words on Hackney Pages') +
  scale_shape_manual(values = 0:10)
```



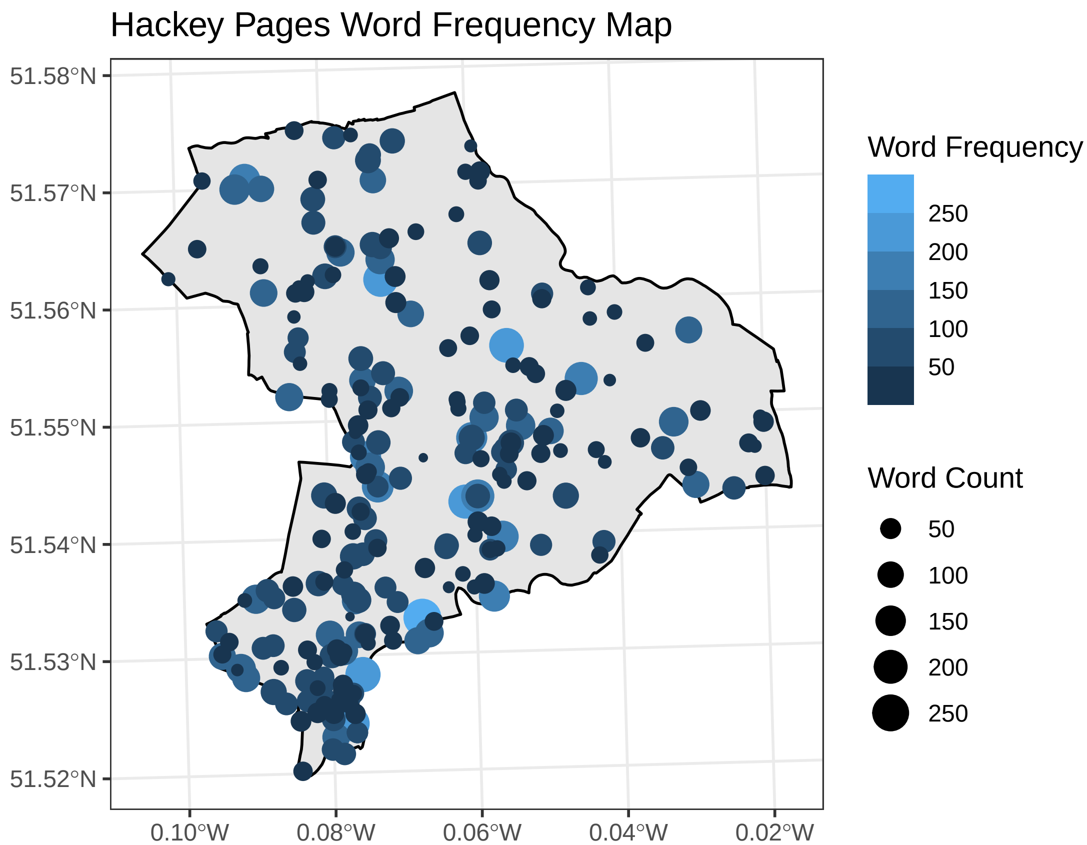
Word Usage Term Frequency Analysis

```
#Top 10 term Frequent words in all Hackney Pages
top_10_hackney_tem_freq_wds <- top_10_hackney_wds %>%
  mutate( term_freq = n/sum(n)) #term frequency

#Map of Top 10 most important word in Hackney based on Term frequency
ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = top_10_hackney_tem_freq_wds,
          aes(color = term_freq, shape = factor(word))) +
  scale_color_steps(name = 'Word Frequency')+
  theme_bw()+
  labs(title = 'Term Frequency: Top 10 Most Important Words') +
  scale_shape_manual(values = 0:10)
```



Total Word Count Per Page Analysis

```
#total number of words on pages
total_wd_on_pages <- page_word_Brtun %>%
  count(page_title,  sort = TRUE)

#Word Frequency map for all Hackney pages
ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = total_wd_on_pages, aes(color = n, size = n)) +
  scale_size(name = "Word Count")+
  scale_color_steps(name = 'Word Frequency')+
  theme_bw()+
  labs(title = 'Hackey Pages Word Frequency Map')
```




```
#Top 10 pages with the highest word frequency
top_10page_wt_hig_wds <- total_wd_on_pages  %>%
  slice_max(n, n=10)

top_10page_wt_hig_wds %>%
  knitr::kable(caption = 'Top 10 Pages with Higest Word Count')
```

Table 3: Top 10 Pages with Higest Word Count

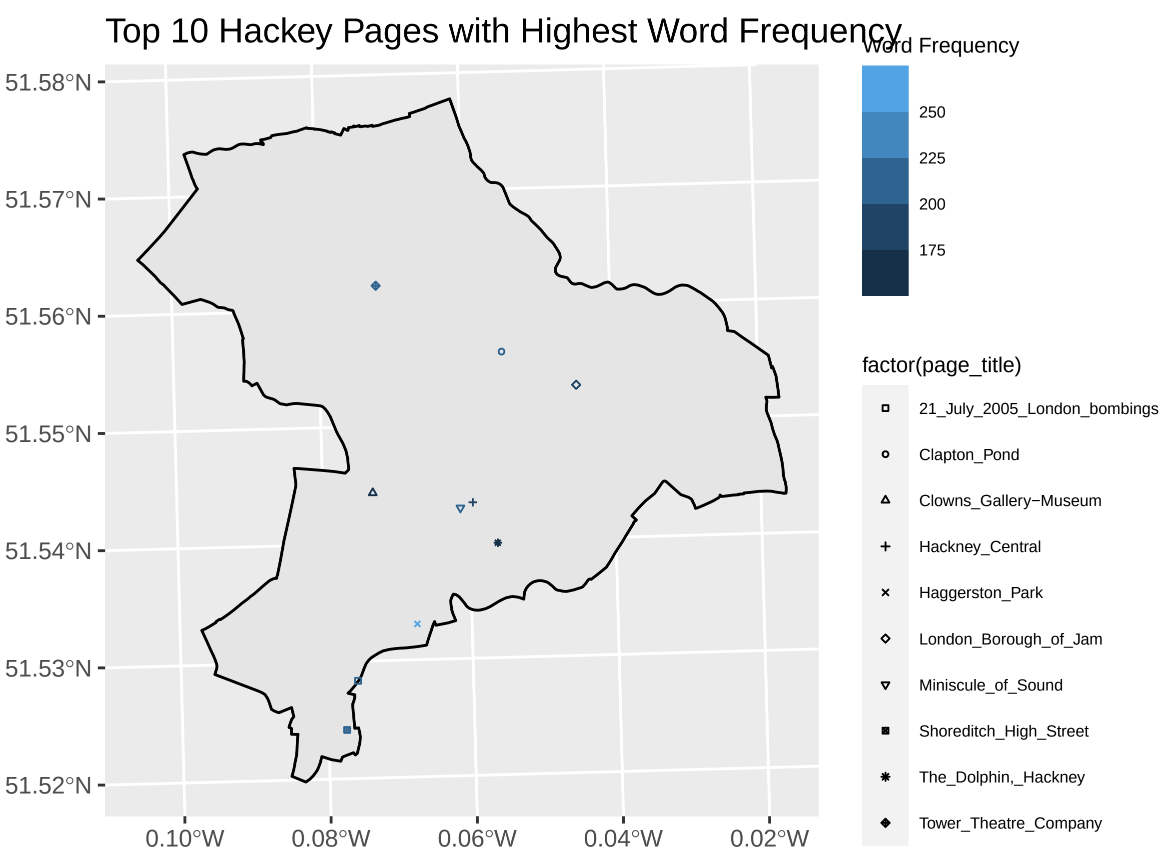
page_title	n	geometry
Haggerston_Park	268	POINT (534137.9 183341.8)
21_July_2005_London_bombings	221	POINT (533573.8 182801.5)
Shoreditch_High_Street	221	POINT (533470.3 182335.1)
Clapton_Pond	213	POINT (534936.7 185927.4)
Tower_Theatre_Company	212	POINT (533741 186552.3)
Miniscule_of_Sound	204	POINT (534546.5 184444.1)
Hackney_Central	190	POINT (534663.5 184496.5)
London_Borough_of_Jam	188	POINT (535645.8 185612.5)
Clowns_Gallery-Museum	167	POINT (533713.8 184585.6)
The_Dolphin,_Hackney	166	POINT (534901.7 184112.6)

```
bottom_10page_wt_hig_wds <- total_wd_on_pages  %>%
  slice_min(n, n=10)

bottom_10page_wt_hig_wds %>%
  knitr::kable(caption = 'Bottom 10 Pages with Lowest Word Count')
```

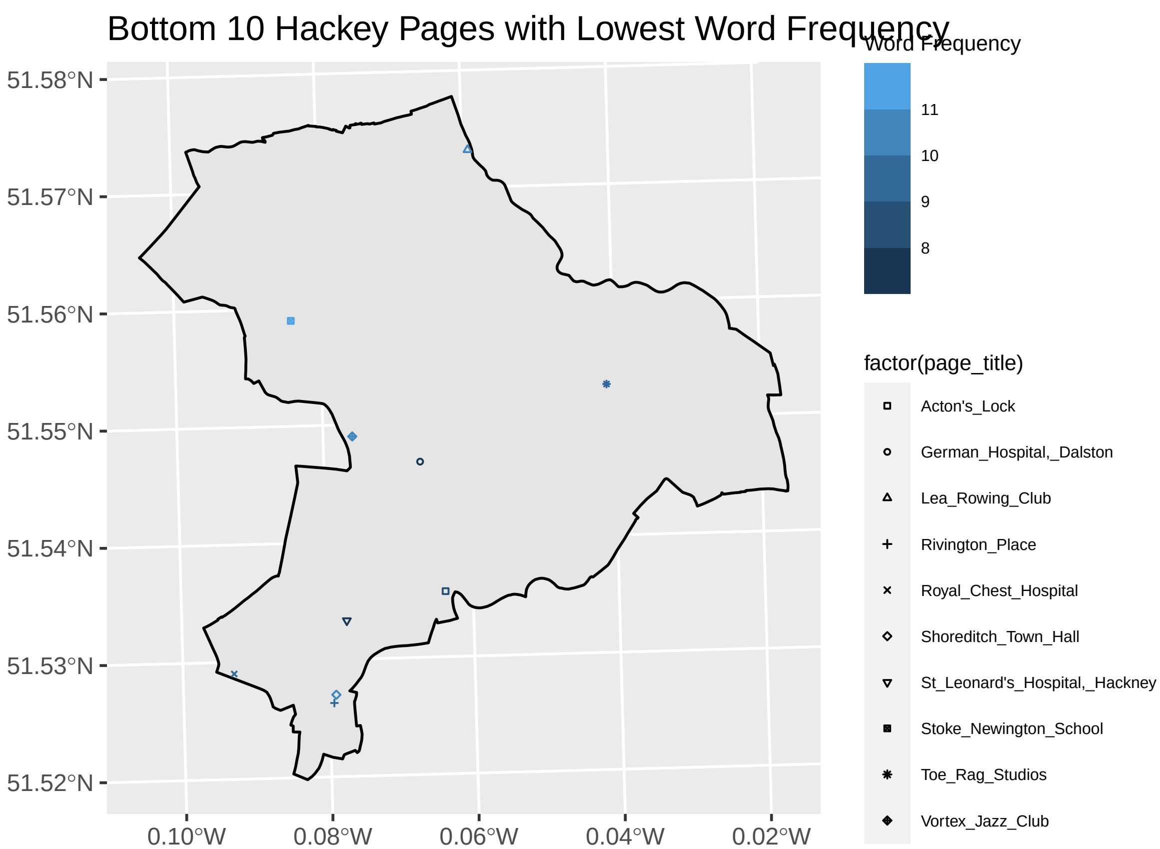
Table 4: Bottom 10 Pages with Lowest Word Count

page_title	n	geometry
German_Hospital,_Dalston	7	POINT (534146.4 184860.4)
St_Leonard's_Hospital,_Hackney	7	POINT (533450.6 183350.9)
Acton's_Lock	9	POINT (534388.6 183630.7)
Rivington_Place	10	POINT (533332.3 182568.9)
Royal_Chest_Hospital	10	POINT (532381.1 182844.4)
Toe_Rag_Studios	10	POINT (535916.7 185597.5)
Lea_Rowing_Club	11	POINT (534596.8 187821)
Shoreditch_Town_Hall	11	POINT (533351.1 182647.2)
Vortex_Jazz_Club	11	POINT (533501.8 185099.4)
Stoke_Newington_School	12	POINT (532918 186196.8)

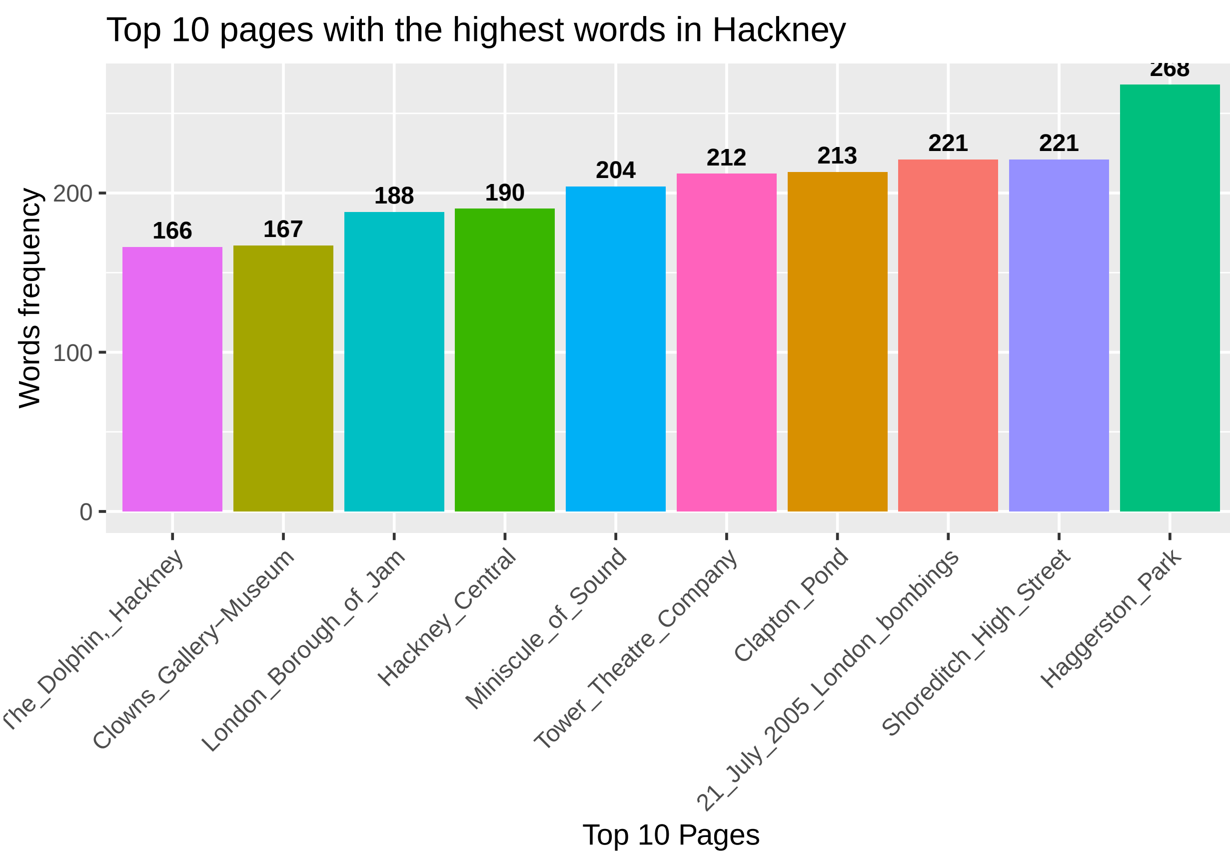
```r
#Map of top 10 Hackney pages with the highest word frequency
ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = top_10page_wt_hig_wds, size = 0.7,
          aes(color = n, shape = factor(page_title ))) +
  scale_color_steps(name = "Word Frequency")+
  theme(legend.title = element_text(size = 8),
        legend.text  = element_text(size = 6))+
  labs(title = 'Top 10 Hackey Pages with Highest Word Frequency') +
  scale_shape_manual(values = 0:10)
```



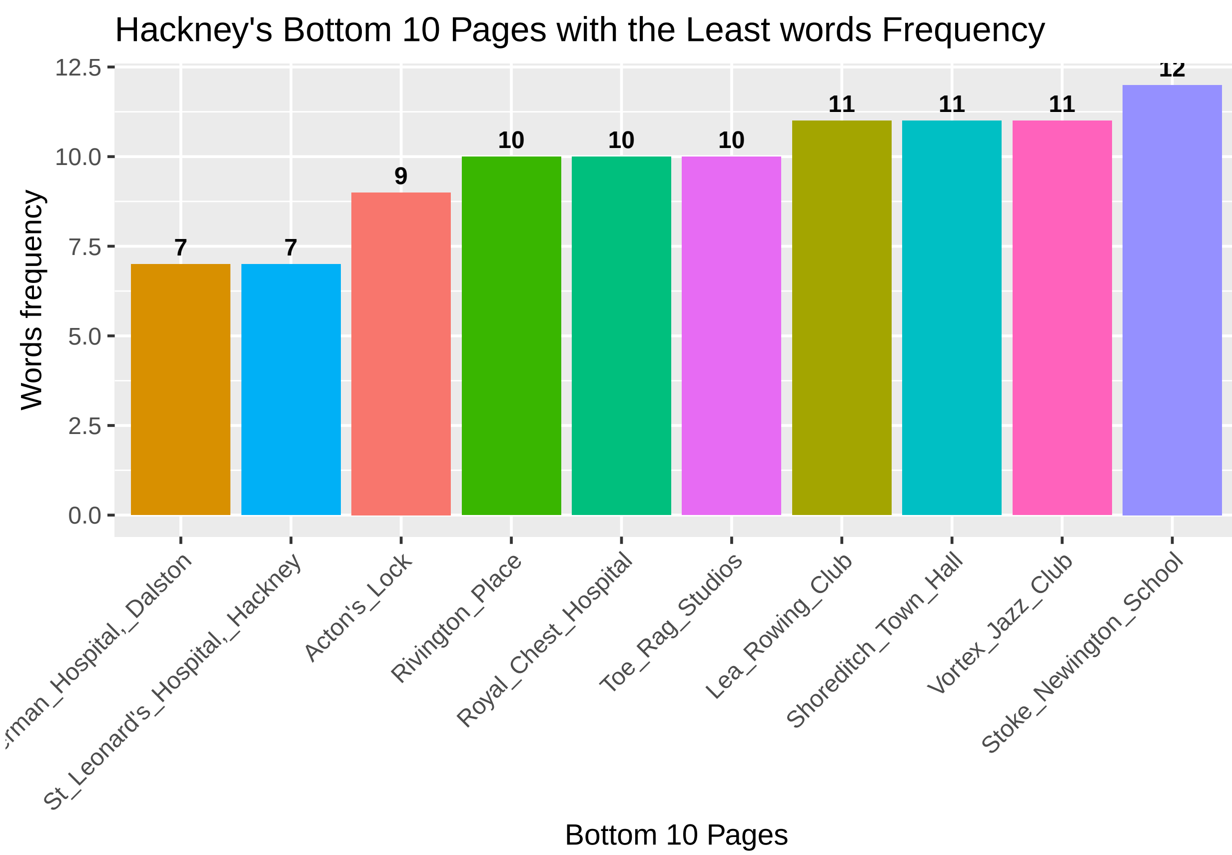
```r
#Map of Bottom 10 Hackney pages with the Lowest word frequency
ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = bottom_10page_wt_hig_wds, size = 0.7,
          aes(color = n, shape = factor(page_title ))) +
  scale_color_steps(name = "Word Frequency")+
  theme(legend.title = element_text(size = 8),
        legend.text  = element_text(size = 6))+
  labs(title = 'Bottom 10 Hackey Pages with Lowest Word Frequency') +
  scale_shape_manual(values = 0:10)
```




```r
#Histogram of Top 10 Hackney pages with the highest word frequency
total_wd_on_pages  %>%
  slice_max(n, n=10) %>%
  ggplot(aes(reorder(page_title, n), n, fill = page_title)) +
  geom_col(show.legend = FALSE) +
  geom_text(aes(label = n), size = 3, fontface = "bold", vjust = -0.5) +
  labs(title = "Top 10 pages with the highest words in Hackney",
       x = "Top 10 Pages", y = "Words frequency") +
  theme(axis.text.x = element_text(angle = 45, hjust=1))
```




```
bottom_10page_wt_hig_wds  %>%
  slice_max(n, n=10) %>%
  ggplot(aes(reorder(page_title, n), n, fill = page_title)) +
  geom_col(show.legend = FALSE) +
  geom_text(aes(label = n), size = 3, fontface = "bold", vjust = -0.5) +
  labs(title = "Hackney's Bottom 10 Pages with the Least words Frequency",
       x = "Bottom 10 Pages", y = "Words frequency") +
  theme(axis.text.x = element_text(angle = 45, hjust=1))
```



Spatial Autocorrelation of words frequency in Hackney Pages

```
#Converting the sf data into sp data to extract coordinates
sp_total_wd_on_pages <- as(total_wd_on_pages, 'Spatial')
coord <- sp::coordinates(sp_total_wd_on_pages)
str(coord)
```

```
##  num [1:241, 1:2] 534138 533574 533470 534937 533741 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "coords.x1" "coords.x2"
```

```
# Creating matrix of points for 1 nearest neighbors
k1 <- knn2nb(knearneigh(coord, k = 1))


# calculating upper bound euclidean distance for atleast 1 neighbor for all points
Eucl_k1dist<- max(unlist(nbdists(k1,coord)))

#Building/defining neighbor points (pages) based on the maximum euclidean distance
sp_total_wd_on_pages.dist  <-  dnearneigh(coord, 0, Eucl_k1dist)

sp_total_wd_on_pages.dist
```

```
## Neighbour list object:
## Number of regions: 241
## Number of nonzero links: 2452
```

```
## Percentage nonzero weights: 4.22169
## Average number of links: 10.17427
```

```
##the result shows that there are 241 points to be linked. The total number of
##connections (neighbors) is 2452, and the average number of links (neighbors) is 10.17


#plotting the neighbor points

plot(hackneyshp$geometry, border='black', lwd=2)
```

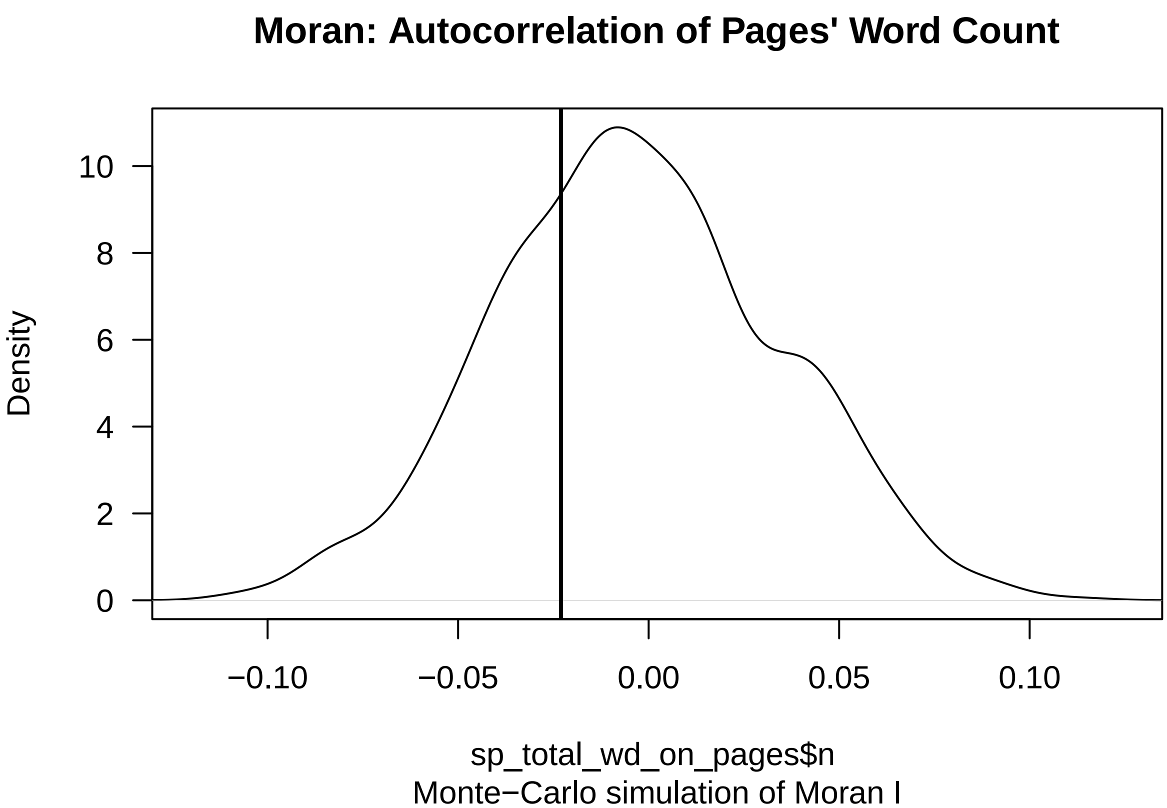
```
plot(sp_total_wd_on_pages.lw, coord, col='brown', lwd=1, add = TRUE)
```

```
## Error in h(simpleError(msg, call)): error in evaluating the argument 'x' in selecting a method for fu
```

```
sp_total_wd_on_pages.lw <- nb2listw(sp_total_wd_on_pages.dist,
                                    style="W",zero.policy=T)

#performing Global Moran's I index with 999 simulations
moran  <-  moran.mc(sp_total_wd_on_pages$n, sp_total_wd_on_pages.lw,
                    nsim=999, zero.policy = T)

print(moran)
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  sp_total_wd_on_pages$n
## weights: sp_total_wd_on_pages.lw
## number of simulations + 1: 1000
##
## statistic = -0.023007, observed rank = 300, p-value = 0.7
## alternative hypothesis: greater
```

```
plot(moran, main="Moran: Autocorrelation of Pages' Word Count", las=1)
```



The p-value of the Moran'I statistics is 0.675.
This indicates that there is about 67% chances of being wrong in rejecting the null hypothesis that there is a random spatial distribution Hackney pages word frequency.
Hence, there is no clustering or specific spatial pattern of Hackney pages word frequency

Word per Page Frequency Analysis

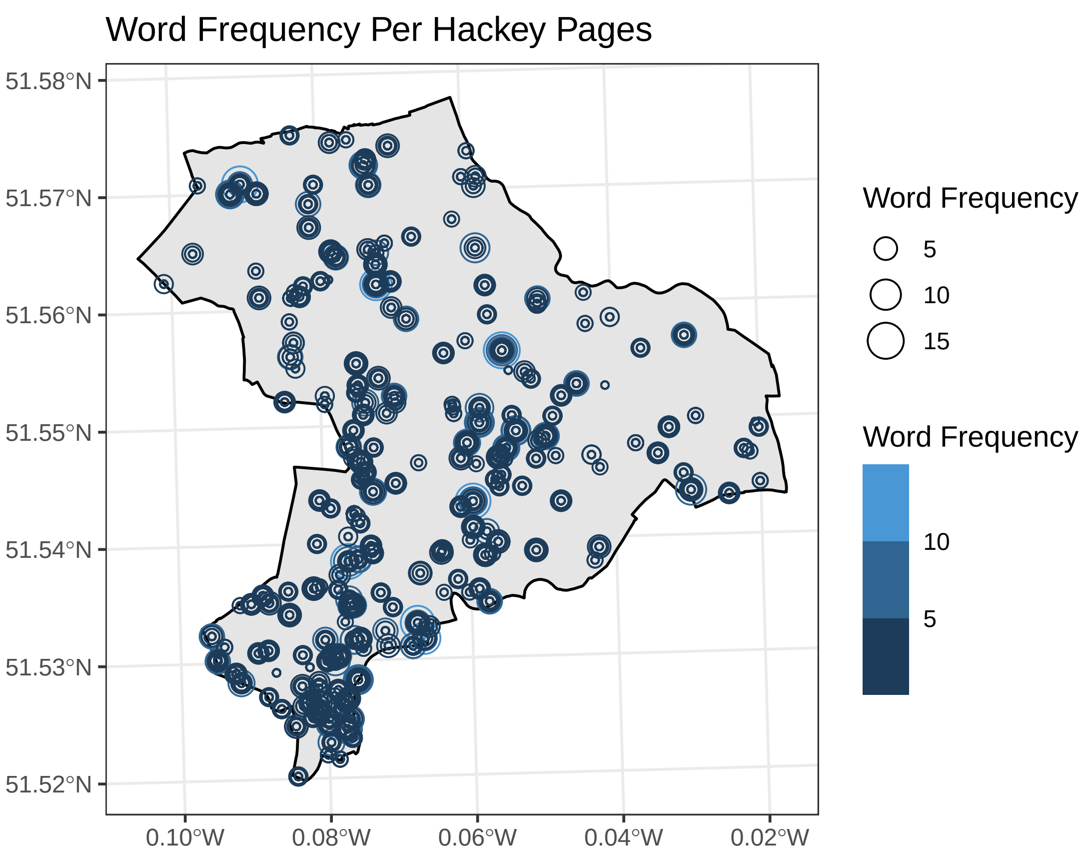
```
#Frequency of each word per page
page_word_count <- page_word_Brtun %>%
  count(page_title, word,  sort = TRUE)


#Top 10 words used per page in
page_word_count %>%
  slice_max(n, n=10) %>%
  knitr::kable(caption = "Top 10 Word-per-Page Frequency")
```

Table 5: Top 10 Word-per-Page Frequency

page_title	word	n	geometry
Clapton_Pond	clapton	15	POINT (534936.7 185927.4)
Woodberry_Down_School	school	15	POINT (532448.5 187501.4)
Hackney_Central	hackney	14	POINT (534663.5 184496.5)
Kingsland_Road	road	14	POINT (533479.9 183923.7)
Haggerston_Park	park	13	POINT (534137.9 183341.8)
Hackney_City_Farm	farm	11	POINT (534203.5 183196.2)
Tower_Theatre_Company	theatre	11	POINT (533741 186552.3)
Clapton_Pond	pond	10	POINT (534936.7 185927.4)

page_title	word	n	geometry
Victoria_Park_railway_station_(England)	station	10	POINT (536734.3 184606.8)
21_July_2005_London_bombings	july	9	POINT (533573.8 182801.5)
Church_of_St_John-at-Hackney	church	9	POINT (535070.8 185165.8)
Clapton_Nursery	nursery	9	POINT (534682 186899.9)
Hackney_College	college	9	POINT (533356.2 182981.2)
Kingsland_Road	kingsland	9	POINT (533479.9 183923.7)
London_Borough_of_Hackney	london	9	POINT (534723.7 185242.9)
Shoreditch_High_Street	shoreditch	9	POINT (533470.3 182335.1)

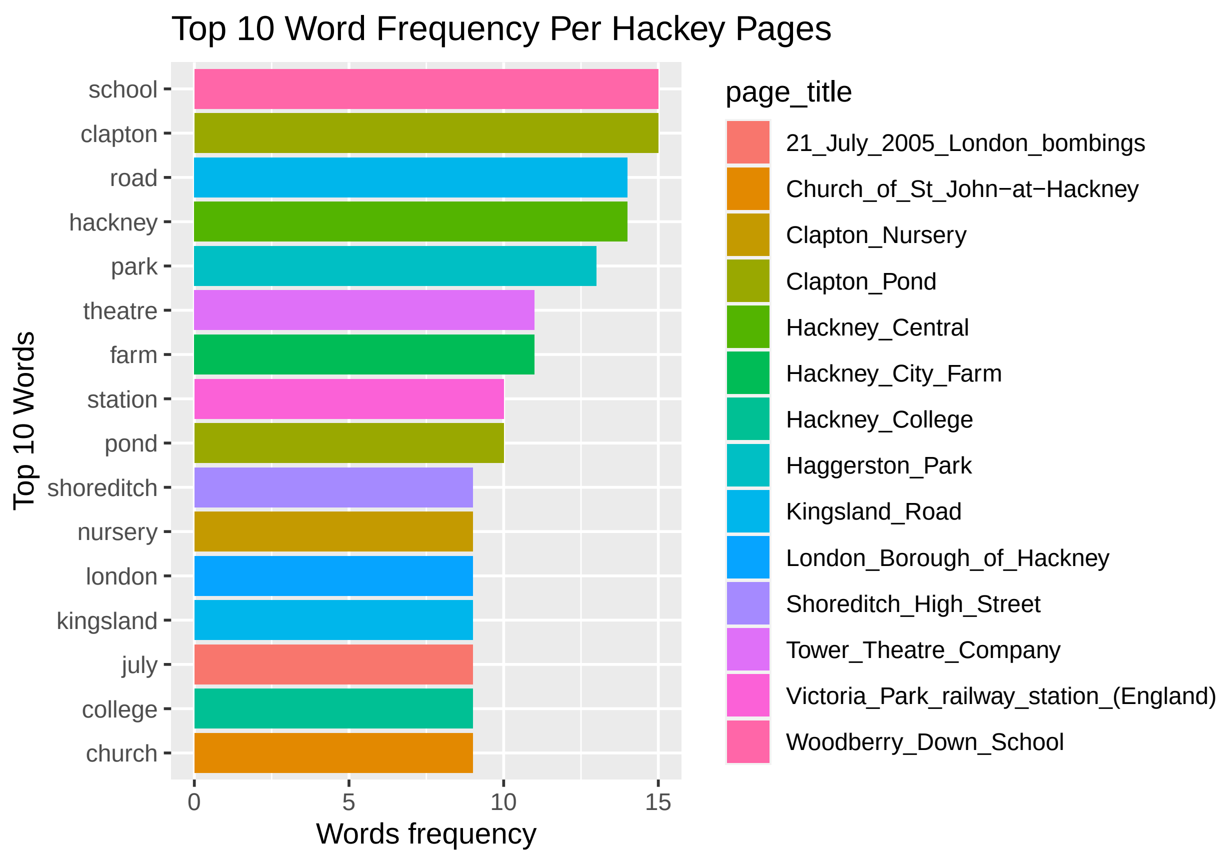
```
#Map of Word Frequency Per Hackney Pages
ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = page_word_count, aes(color = n, size = n),
          shape = factor(page_title )) +
  scale_size(name = "Word Frequency")+
  scale_color_steps(name = 'Word Frequency')+
  theme_bw()+
  labs(title = 'Word Frequency Per Hackey Pages')
```



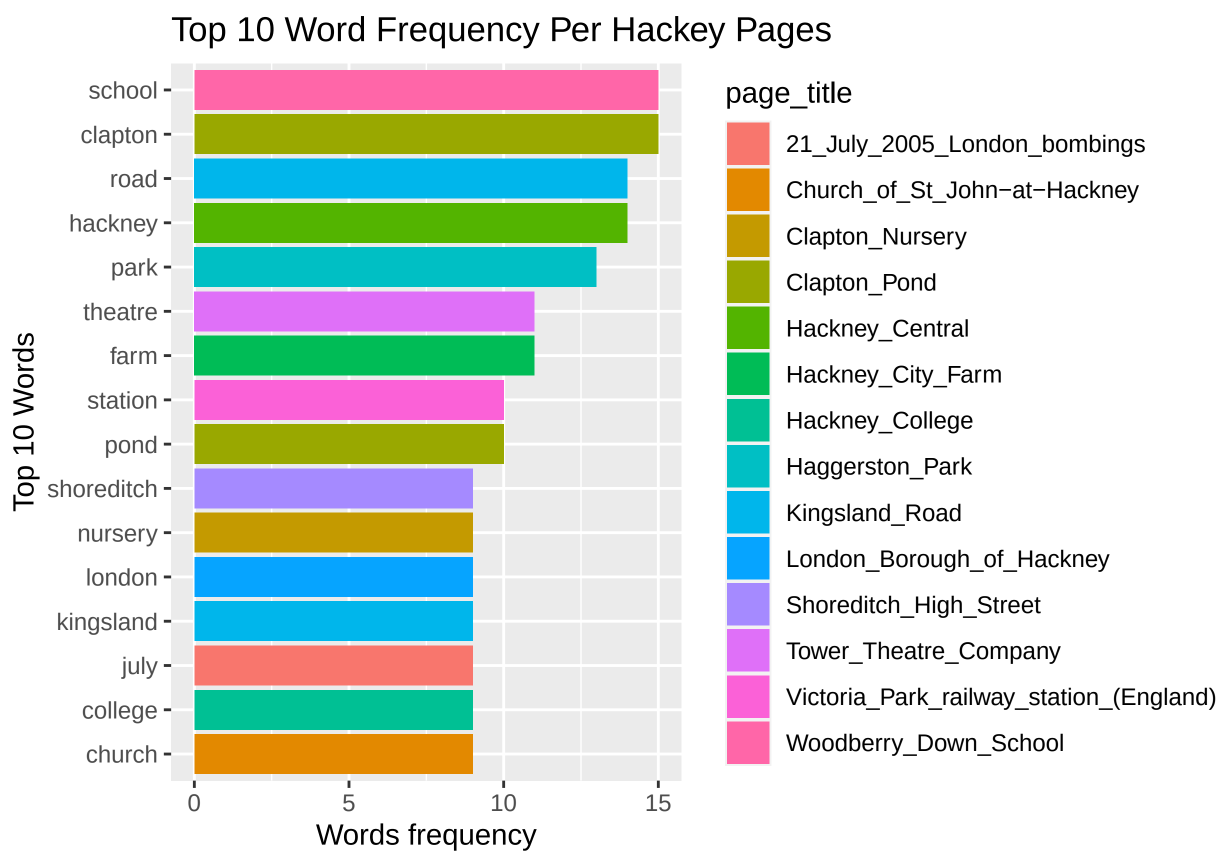
Top 10 Words Frequency per Hackney Pages Analysis

```
#Histogram of Top 10 Word Frequency Per Hackney Pages
page_word_count %>%
  slice_max(n, n=10) %>%
```

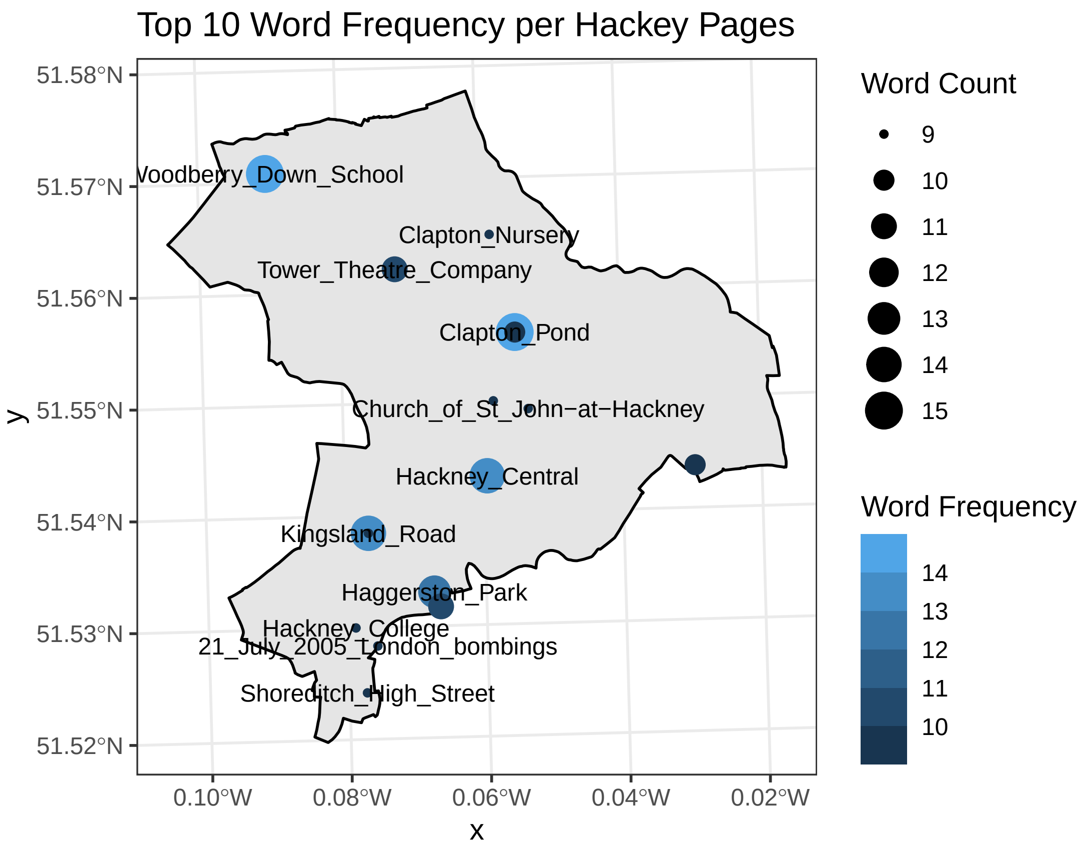
```
  mutate(ordr = reorder(word, n)) %>%
  ggplot(aes(ordr, n, fill = page_title)) +
  geom_col() +
  labs(title = 'Top 10 Word Frequency Per Hackey Pages',
       x = "Top 10 Words", y = "Words frequency") +
  coord_flip()
```



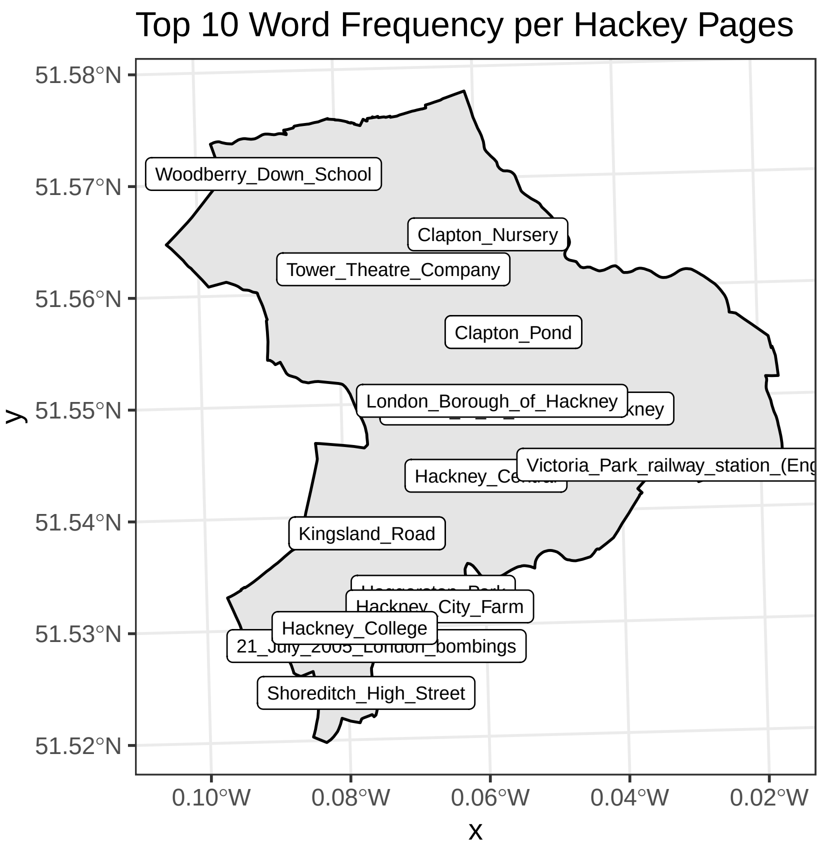

```
top10_page_word_count <- page_word_count %>%
  slice_max(n, n=10)

#Histogram of Top 10 Word Frequency Per Hackney Pages
page_word_count %>%
  slice_max(n, n=10) %>%
  mutate(ordr = reorder(word, n)) %>%
  ggplot(aes(ordr, n, fill = page_title)) +
  geom_col() +
  labs(title = 'Top 10 Word Frequency Per Hackey Pages',
       x = "Top 10 Words", y = "Words frequency") +
  coord_flip()
```




```r
top10_page_word_count <- page_word_count %>%
  slice_max(n, n=10)

#Map of Top 10 Word Frequency Per Hackney Pages
ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = top10_page_word_count, aes(color = n, size = n)) +
  scale_size(name = "Word Count")+
  scale_color_steps(name = 'Word Frequency')+
  geom_sf_text(data = top10_page_word_count, aes(label = page_title), size = 3,
               color = 'black', check_overlap = T)+
  theme_bw()+
  labs(title = 'Top 10 Word Frequency per Hackey Pages ')
```




```r
ggplot(top10_page_word_count) +
  geom_sf(aes(color = n), show.legend = FALSE) +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf_label(aes(label = page_title), size = 2.5)+
  theme_bw()+
  labs(title = 'Top 10 Word Frequency per Hackey Pages ')
```



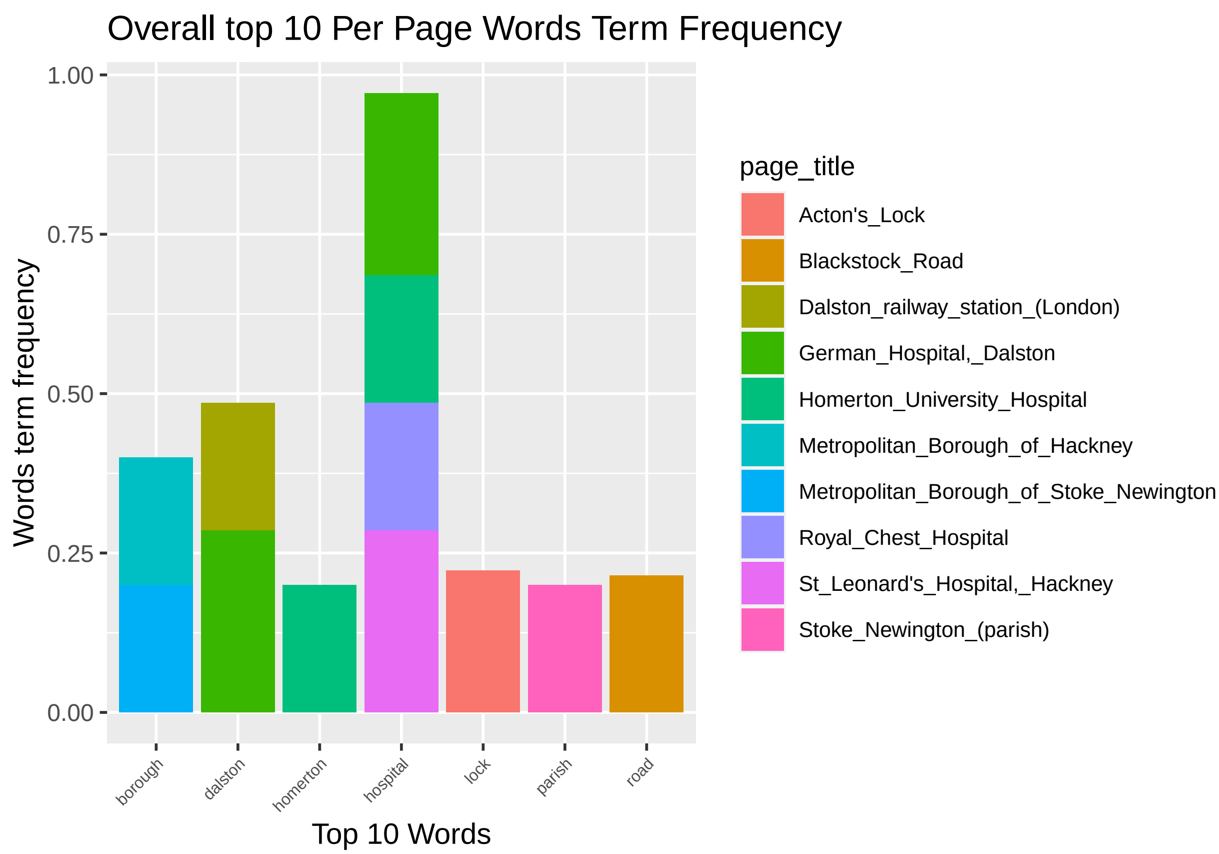
Term Frequency Analysis for Word per Page Frequency

```
#Top 10 Normalized (term) frequency of words per page title
page_word_count %>%
  group_by(page_title) %>%
  mutate( term_freq = n/sum(n)) %>%
  ungroup() %>%
  slice_max(term_freq, n=10) %>%
  knitr::kable(caption = "Term Frequency: Top 10 Most Important Word-per-Page")
```

Table 6: Term Frequency: Top 10 Most Important Word-per-Page

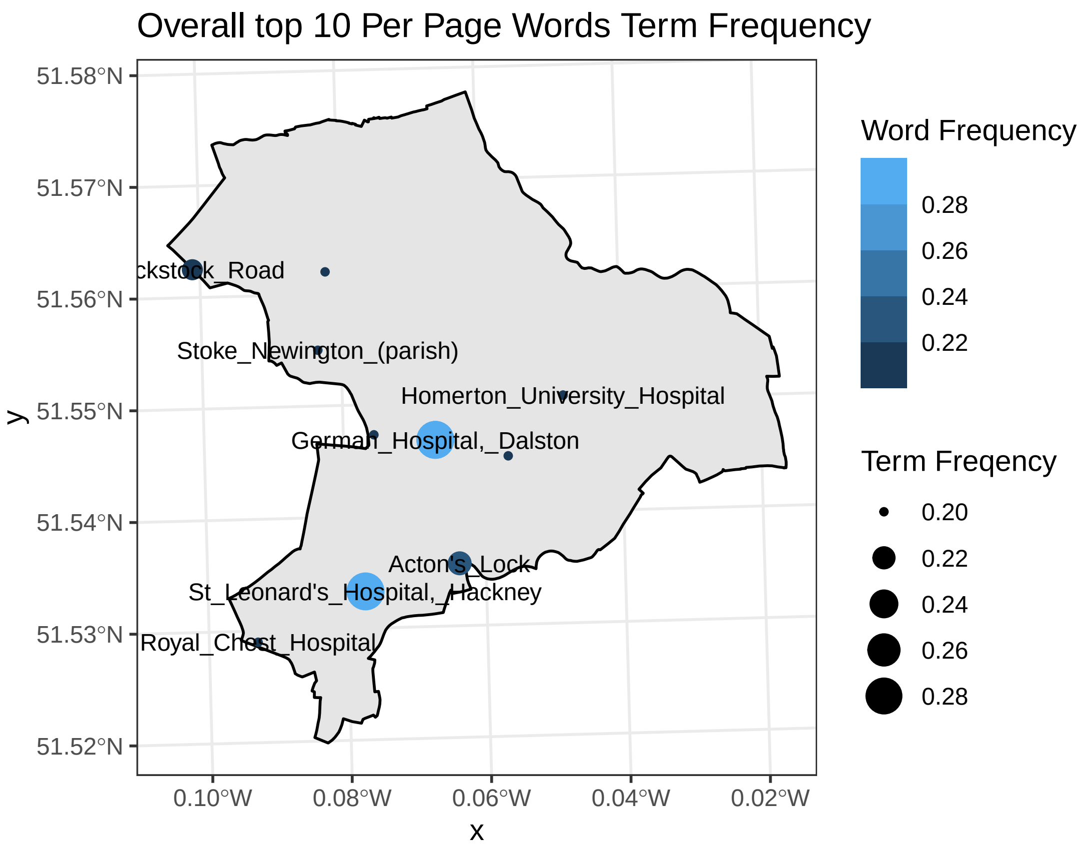
page_title	word	n	geometry	term_freq
German_Hospital,_Dalston	dalston	2	POINT (534146.4 184860.4)	0.2857143
German_Hospital,_Dalston	hospital	2	POINT (534146.4 184860.4)	0.2857143
St_Leonard's_Hospital,_Hackney	hospital	2	POINT (533450.6 183350.9)	0.2857143
Acton's_Lock	lock	2	POINT (534388.6 183630.7)	0.2222222
Blackstock_Road	road	3	POINT (531726 186555.2)	0.2142857
Dalston_railway_station_(London)	dalston	4	POINT (533534.5 184911.1)	0.2000000
Homerton_University_Hospital	homerton	3	POINT (535418.1 185305.9)	0.2000000
Homerton_University_Hospital	hospital	3	POINT (535418.1 185305.9)	0.2000000
Metropolitan_Borough_of_Hackney	borough	3	POINT (534872.2 184701.6)	0.2000000
Metropolitan_Borough_of_Stoke_Newington	borough	3	POINT (533047.9 186534.1)	0.2000000
Stoke_Newington_(parish)	parish	3	POINT (532975.8 185753.2)	0.2000000
Royal_Chest_Hospital	hospital	2	POINT (532381.1 182844.4)	0.2000000

```r
#Overall top 10 Per page word term frequency Histogram
##This section shows the term frequency of each words per
#their individual pages word frequency,
## and the overall top ten of these term frequency score.
page_word_count %>%
  group_by(page_title) %>%
  mutate( term_freq = n/sum(n)) %>% #term frequency
  ungroup() %>%
  slice_max(term_freq, n = 10) %>%
  ggplot(aes(word, term_freq, fill = page_title)) +
  geom_col() +
  theme(legend.title = element_text(size = 10),
        legend.text  = element_text(size = 8),
        axis.text.x = element_text(angle = 45, hjust = 1, size = 6)) +
  labs(title = "Overall top 10 Per Page Words Term Frequency",
       y = "Words term frequency", x= 'Top 10 Words')
```

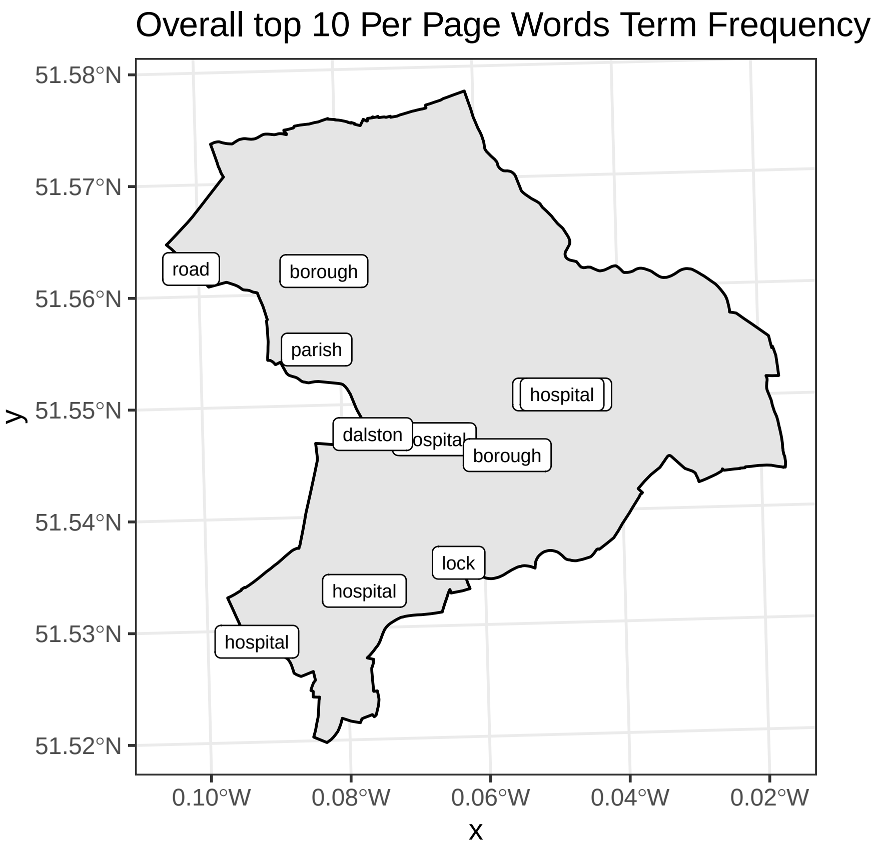



```r
top10temfreqppage <- page_word_count %>%
  group_by(page_title) %>%
  mutate( term_freq = n/sum(n)) %>% #term frequency
  ungroup() %>%
  slice_max(term_freq, n = 10)

#Overall top 10 Per page word term frequency map
ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = top10temfreqppage, aes(color = term_freq, size = term_freq)) +
```

```r
  scale_size(name = "Term Freqency")+
  scale_color_steps(name = 'Word Frequency')+
  geom_sf_text(data = top10temfreqppage, aes(label = page_title), size = 3,
               color = 'black', check_overlap = T)+
  theme_bw()+
  labs(title = 'Overall top 10 Per Page Words Term Frequency')
```




```r
ggplot(top10temfreqppage) +
  geom_sf(aes(color = term_freq), show.legend = FALSE) +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf_label(aes(label = word), size = 2.5)+
  theme_bw()+
  labs(title = 'Overall top 10 Per Page Words Term Frequency')
```



Term Frequency Analysis per Overall Word count

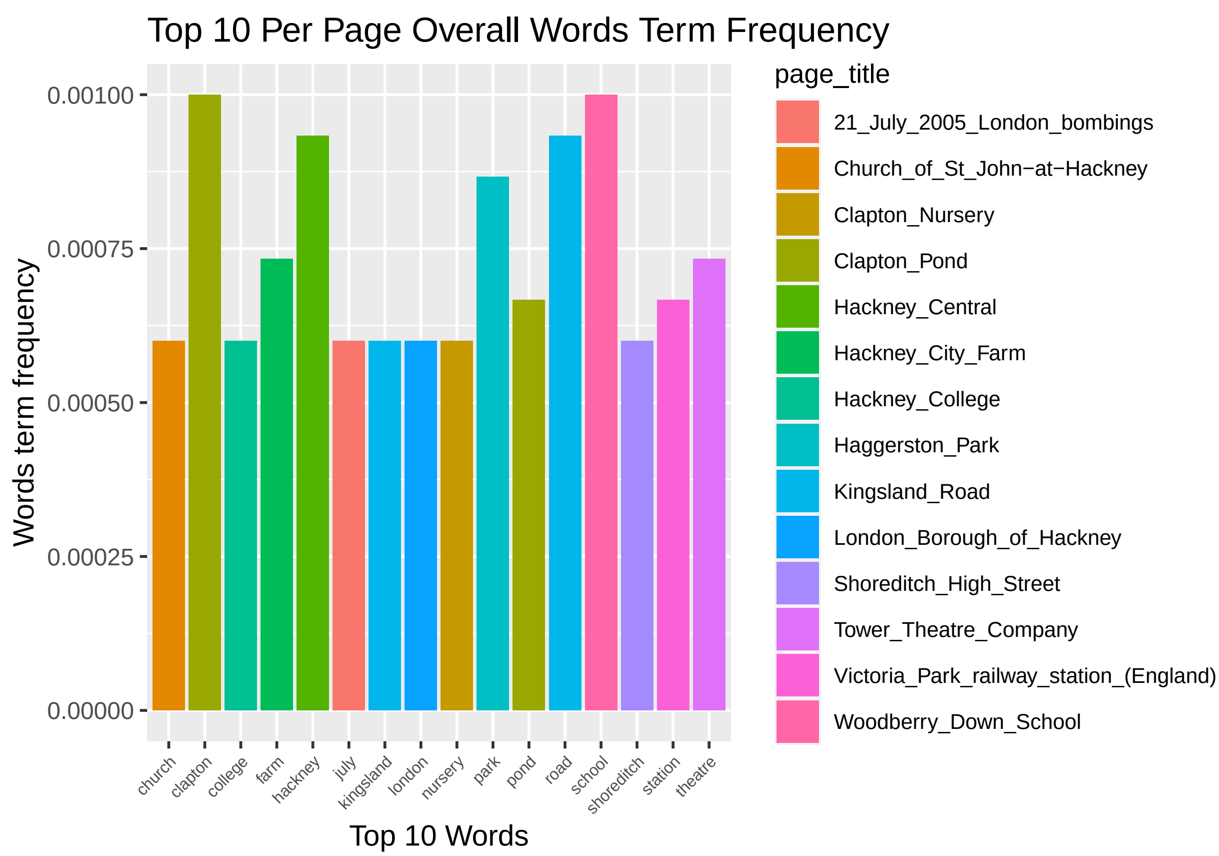
```
#Top 10 per page work term frequency
#This section shows the per page term frequency of each word based on the
#overall word count in all Hackney pages
page_word_count %>%
  mutate( term_freq = n/sum(n)) %>%
  slice_max(term_freq, n=10) %>%
  knitr::kable(caption = "Term Frequency: Overall Top 10 Most Important Words")
```

Table 7: Term Frequency: Overall Top 10 Most Important Words

page_title	word	n	term_freq	geometry
Clapton_Pond	clapton	15	0.0010000	POINT (534936.7 185927.4)
Woodberry_Down_School	school	15	0.0010000	POINT (532448.5 187501.4)
Hackney_Central	hackney	14	0.0009333	POINT (534663.5 184496.5)
Kingsland_Road	road	14	0.0009333	POINT (533479.9 183923.7)
Haggerston_Park	park	13	0.0008667	POINT (534137.9 183341.8)
Hackney_City_Farm	farm	11	0.0007333	POINT (534203.5 183196.2)
Tower_Theatre_Company	theatre	11	0.0007333	POINT (533741 186552.3)
Clapton_Pond	pond	10	0.0006667	POINT (534936.7 185927.4)
Victoria_Park_railway_station_(England)	station	10	0.0006667	POINT (536734.3 184606.8)
21_July_2005_London_bombings	july	9	0.0006000	POINT (533573.8 182801.5)
Church_of_St_John-at-Hackney	church	9	0.0006000	POINT (535070.8 185165.8)
Clapton_Nursery	nursery	9	0.0006000	POINT (534682 186899.9)
Hackney_College	college	9	0.0006000	POINT (533356.2 182981.2)

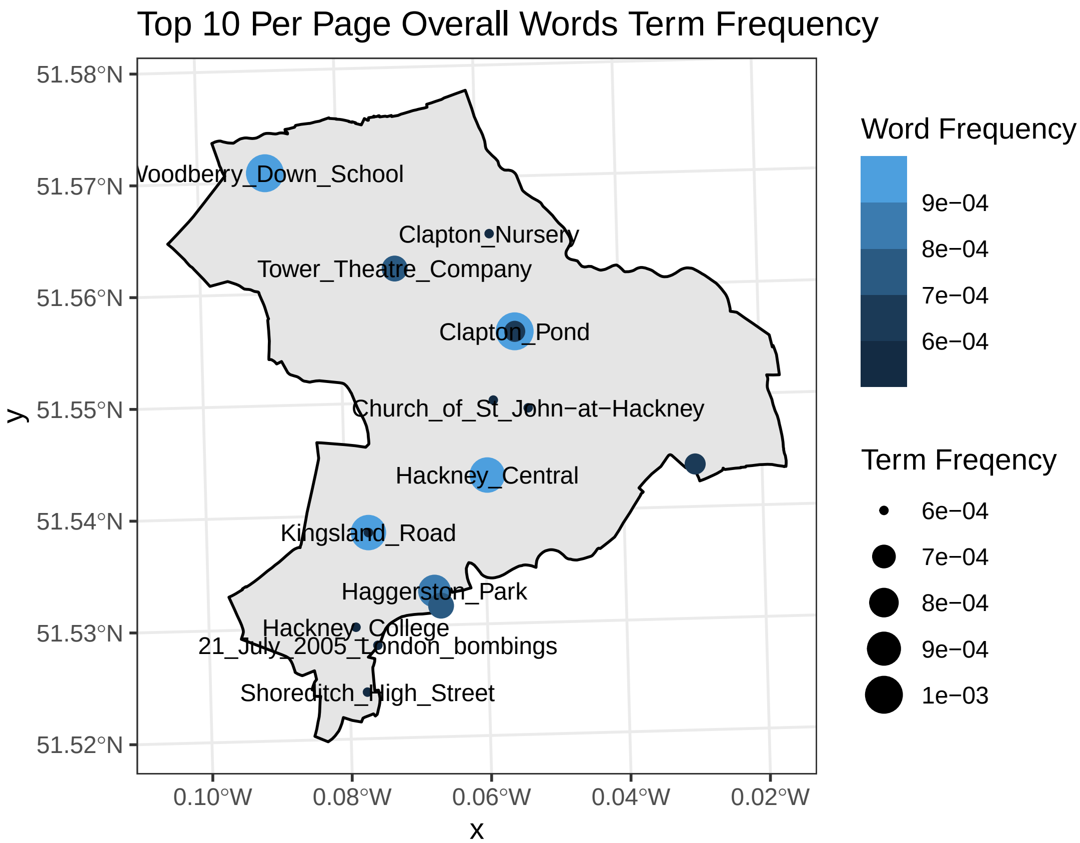
page_title	word	n	term_freq	geometry
Kingsland_Road	kingsland	9	0.0006000	POINT (533479.9 183923.7)
London_Borough_of_Hackney	london	9	0.0006000	POINT (534723.7 185242.9)
Shoreditch_High_Street	shoreditch	9	0.0006000	POINT (533470.3 182335.1)

```r
temfreqppage_top10 <- page_word_count %>%
  mutate( term_freq = n/sum(n)) %>%
  slice_max(term_freq, n=10)

#Histogram of top 10 Per Page Overall Words Term Frequency
temfreqppage_top10 %>%
  ggplot(aes(word, term_freq, fill = page_title)) +
  geom_col() +
  theme(legend.title = element_text(size = 10),
        legend.text  = element_text(size = 8),
        axis.text.x = element_text(angle = 45, hjust = 1, size = 6)) +
  labs(title = "Top 10 Per Page Overall Words Term Frequency",
       y = "Words term frequency", x= 'Top 10 Words')
```

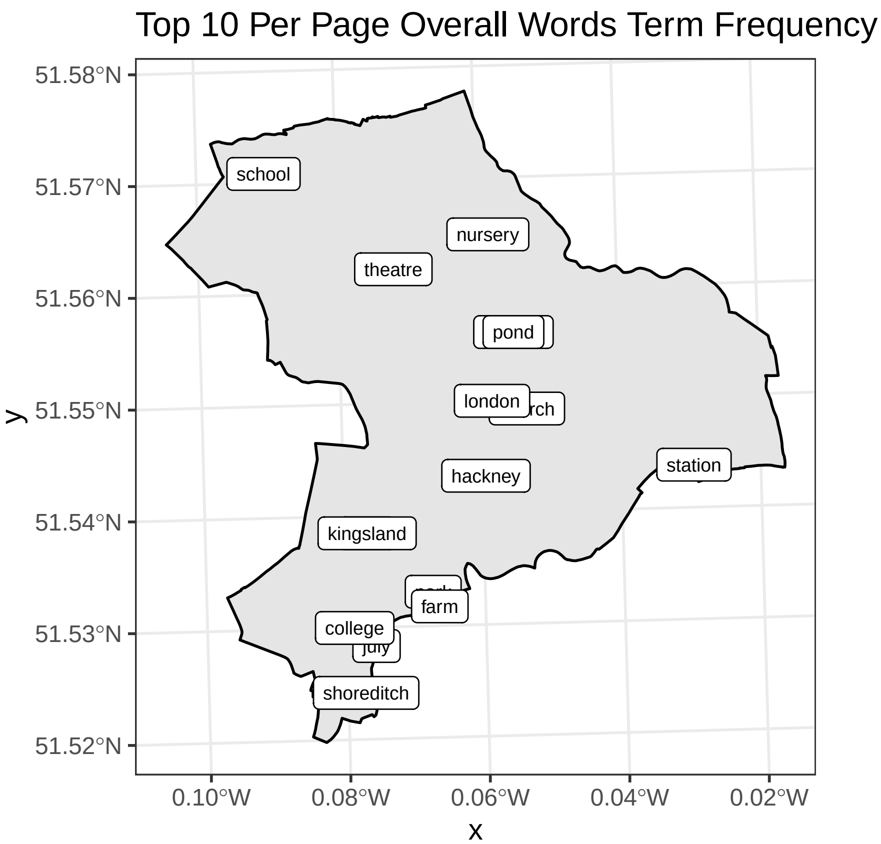



```r
##Top 10 Per Page Overall Words Term Frequency map
ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = temfreqppage_top10, aes(color = term_freq, size = term_freq)) +
  scale_size(name = "Term Freqency")+
  scale_color_steps(name = 'Word Frequency')+
  geom_sf_text(data = temfreqppage_top10, aes(label = page_title), size = 3,
```

```
                color = 'black', check_overlap = T)+
  theme_bw()+
  labs(title = 'Top 10 Per Page Overall Words Term Frequency')
```




```
ggplot(temfreqppage_top10) +
  geom_sf(aes(color = term_freq), show.legend = FALSE) +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf_label(aes(label = word), size = 2.5)+
  theme_bw()+
  labs(title = 'Top 10 Per Page Overall Words Term Frequency')
```



Tfidf Frequency Analysis

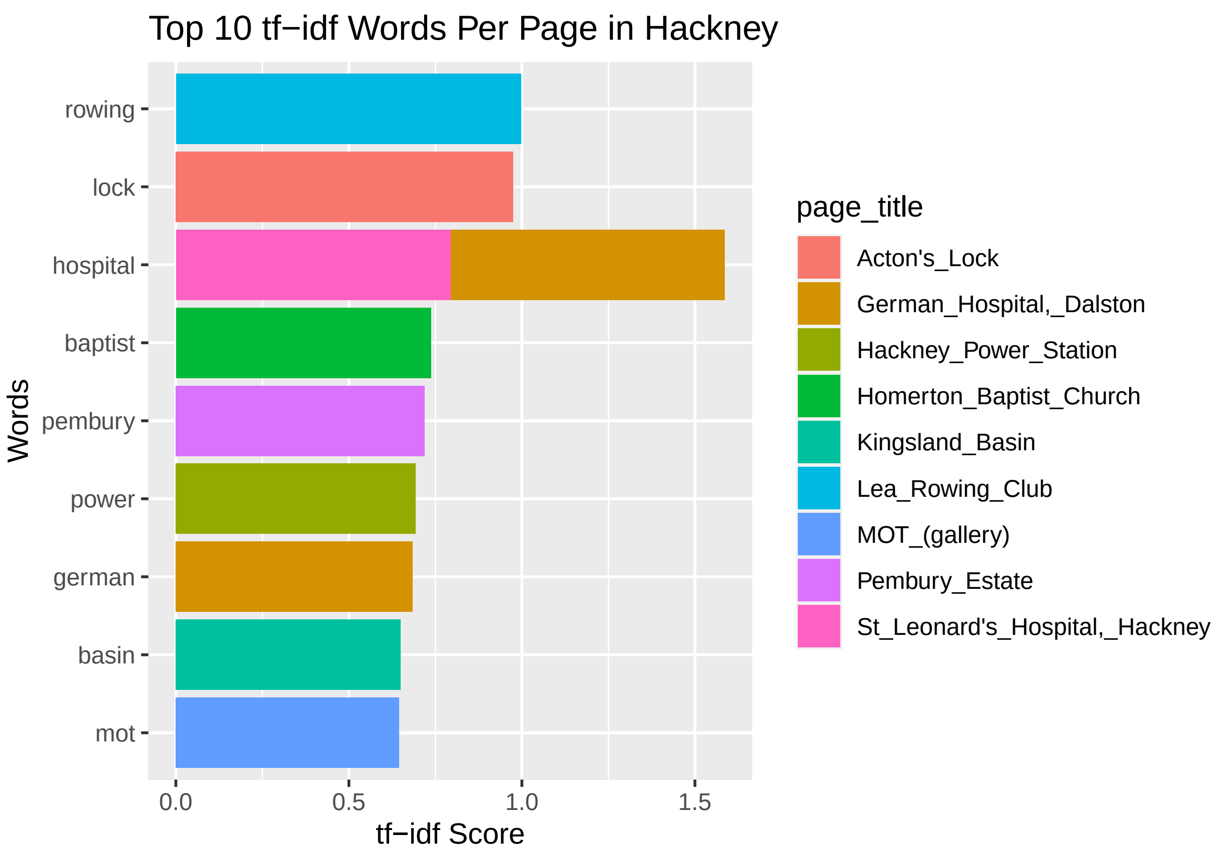
```r
#Top 10 Tfidf score of words per page
page_word_count %>%
  bind_tf_idf(word, page_title, n) %>%
  slice_max(tf_idf, n=10) %>%
  knitr::kable(caption = "Tf-idf: Top 10 most Important Words")
```

Table 8: Tf-idf: Top 10 most Important Words

page_title	word	n	tf	idf	tf_idf	geometry
Lea_Rowing_Club	rowing	2	0.1818182	5.484797	0.9972358	POINT (534596.8 187821)
Acton's_Lock	lock	2	0.2222222	4.386185	0.9747077	POINT (534388.6 183630.7)
German_Hospital,_Dalston	hospital	2	0.2857143	2.776747	0.7933562	POINT (534146.4 184860.4)
St_Leonard's_Hospital,_Hackney	hospital	2	0.2857143	2.776747	0.7933562	POINT (533450.6 183350.9)
Homerton_Baptist_Church	baptist	2	0.1538462	4.791650	0.7371769	POINT (535869.7 184820.8)
Pembury_Estate	pembury	8	0.1311475	5.484797	0.7193176	POINT (534725.2 185382.4)
Hackney_Power_Station	power	3	0.1578947	4.386185	0.6925555	POINT (535961.8 186244.1)

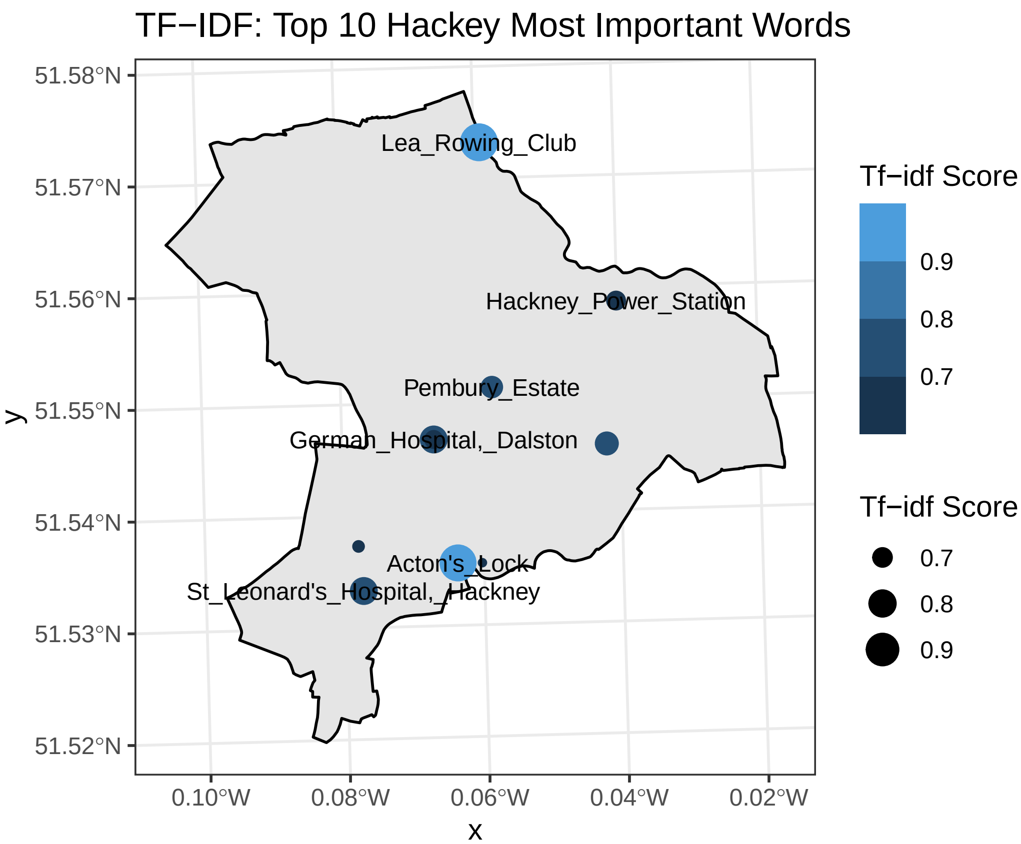
page_title	word	n	tf	idf	tf_idf	geometry
German_Hospital,_Dalston	german	1	0.1428571	4.791650	0.6845214	POINT (534146.4 184860.4)
Kingsland_Basin	basin	4	0.1481481	4.386185	0.6498051	POINT (533398 183795)
MOT_(gallery)	mot	2	0.1176471	5.484797	0.6452702	POINT (534631.4 183633.1)

```r
page_word_count %>%
  bind_tf_idf(word, page_title, n) %>%
  slice_max(tf_idf, n=10) %>%
  ggplot(aes(tf_idf, fct_reorder(word, tf_idf), fill = page_title)) +
  geom_col() +
  labs(title = 'Top 10 tf-idf Words Per Page in Hackney', x = 'tf-idf Score', y = 'Words')
```

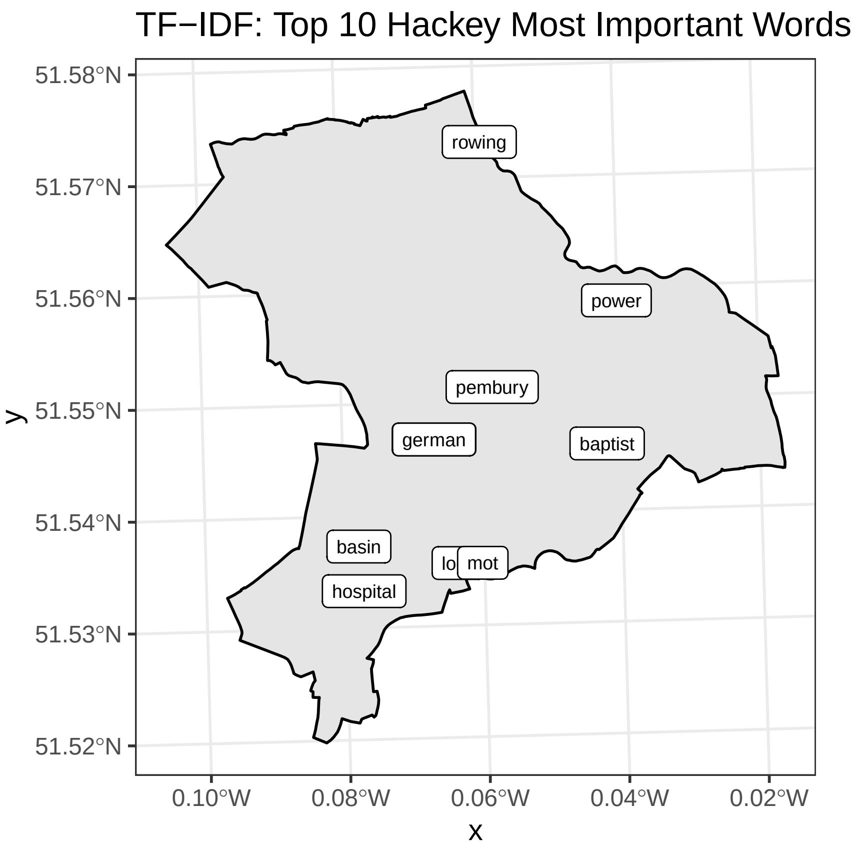



```r
top10_tfidfppage <- page_word_count %>%
  bind_tf_idf(word, page_title, n) %>%
  slice_max(tf_idf, n=10)

ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = top10_tfidfppage, aes(color = tf_idf, size = tf_idf)) +
  scale_size(name = "Tf-idf Score")+
  scale_color_steps(name = 'Tf-idf Score')+
  geom_sf_text(data = top10_tfidfppage, aes(label = page_title), size = 3,
               color = 'black', check_overlap = T)+
  theme_bw()+
```

```r
  labs(title =  'TF-IDF: Top 10 Hackey Most Important Words')
```




```r
ggplot(top10_tfidfppage) +
  geom_sf(aes(color = tf_idf), show.legend = FALSE) +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf_label(aes(label = word), size = 2.5)+
  theme_bw()+
  labs(title = 'TF-IDF: Top 10 Hackey Most Important Words')
```



Weighted Log Frequency Analysis

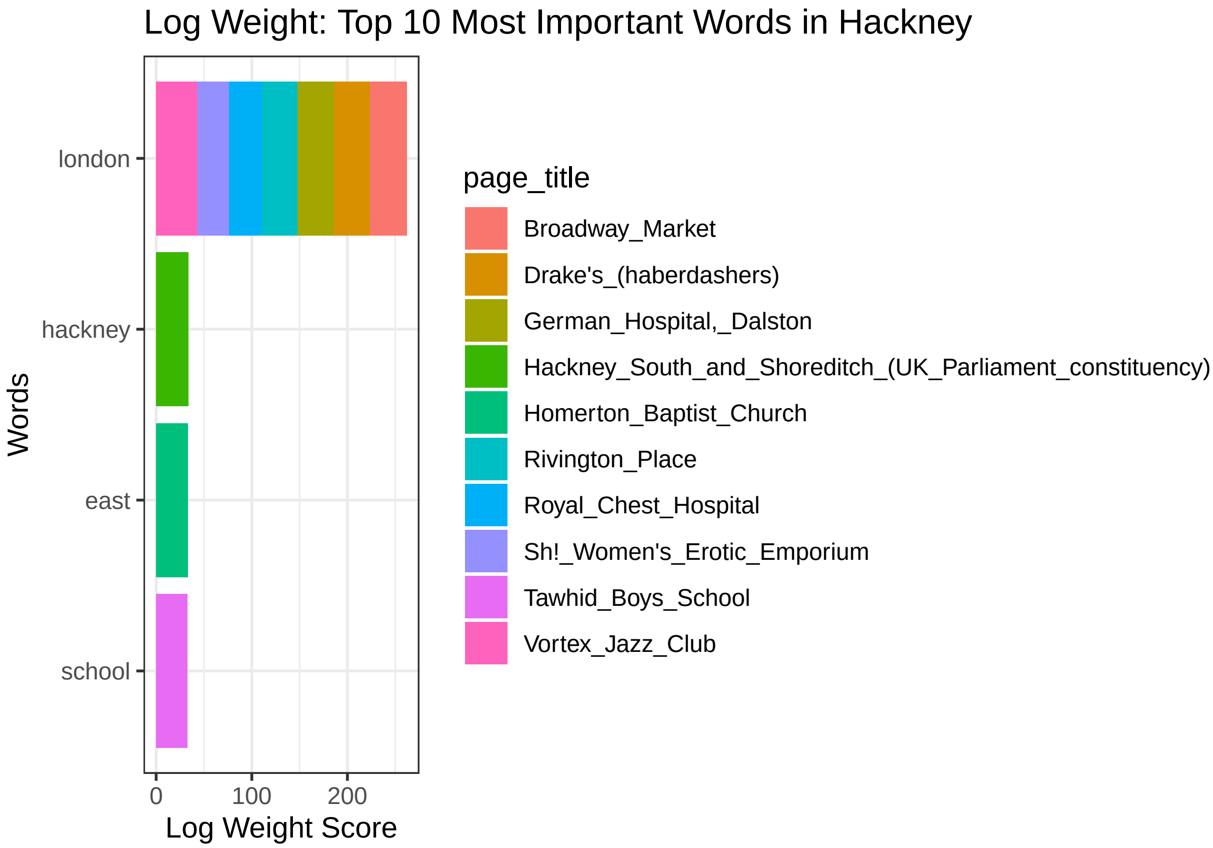
```r
#Weighted Log
top10weightlog <- page_word_count %>%
  st_drop_geometry() %>%
  bind_log_odds(page_title, word, n) %>%
  # group_by(page_title) %>%
  # slice_max(log_odds_weighted) %>%
  # ungroup() %>%
  slice_max(log_odds_weighted, n =10)

top10weightlog%>%
  knitr::kable(caption = "Weighted Log: Top 10 most Important Words")
```

Table 9: Weighted Log: Top 10 most Important Words

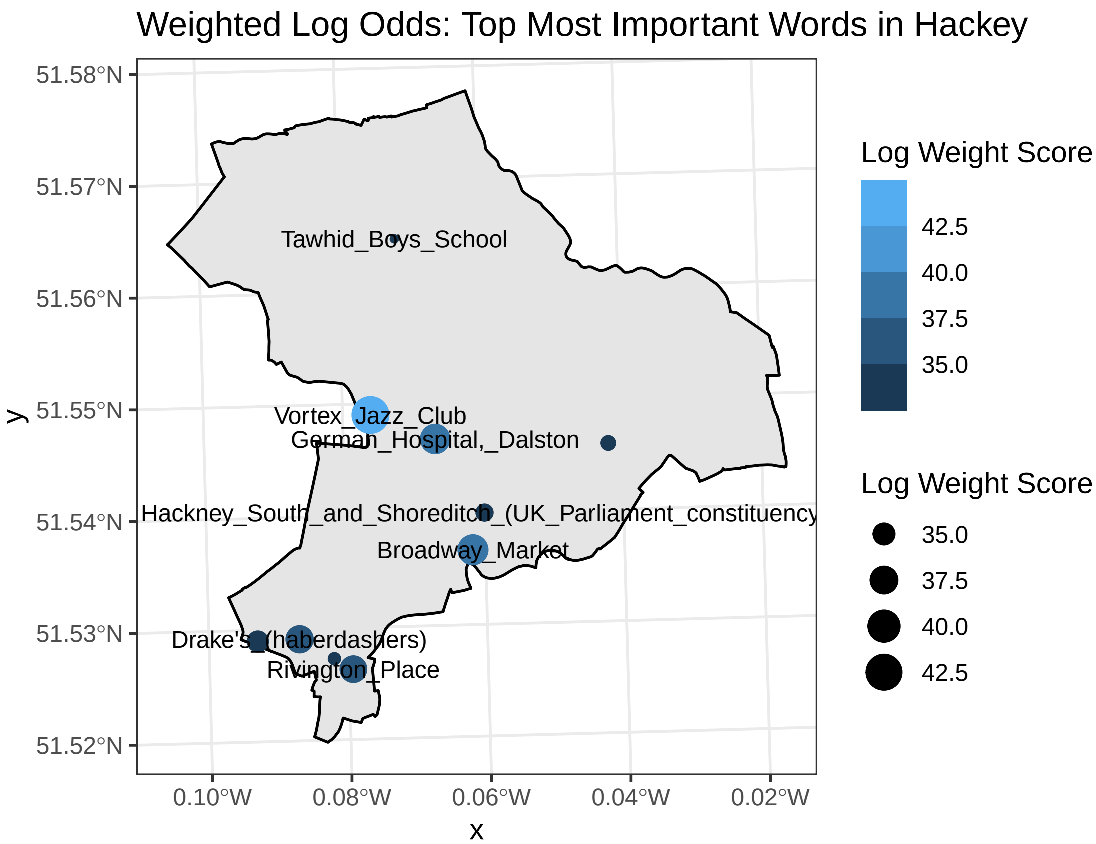
page_title	word	n	log_odds_weighted
Vortex_Jazz_Club	london	1	43.14973
Broadway_Market	london	1	38.71651
German_Hospital,_Dalston	london	1	38.39462
Drake's_(haberdashers)	london	1	37.25453
Rivington_Place	london	1	36.95396
Royal_Chest_Hospital	london	1	34.69152
Hackney_South_and_Shoreditch_(UK_Parliament_constituency)	hackney	1	33.60843
Homerton_Baptist_Church	east	1	33.13641
Sh!_Women's_Erotic_Emporium	london	1	32.81050

page_title	word	n	log_odds_weighted
Tawhid_Boys_School	school	6	32.60265

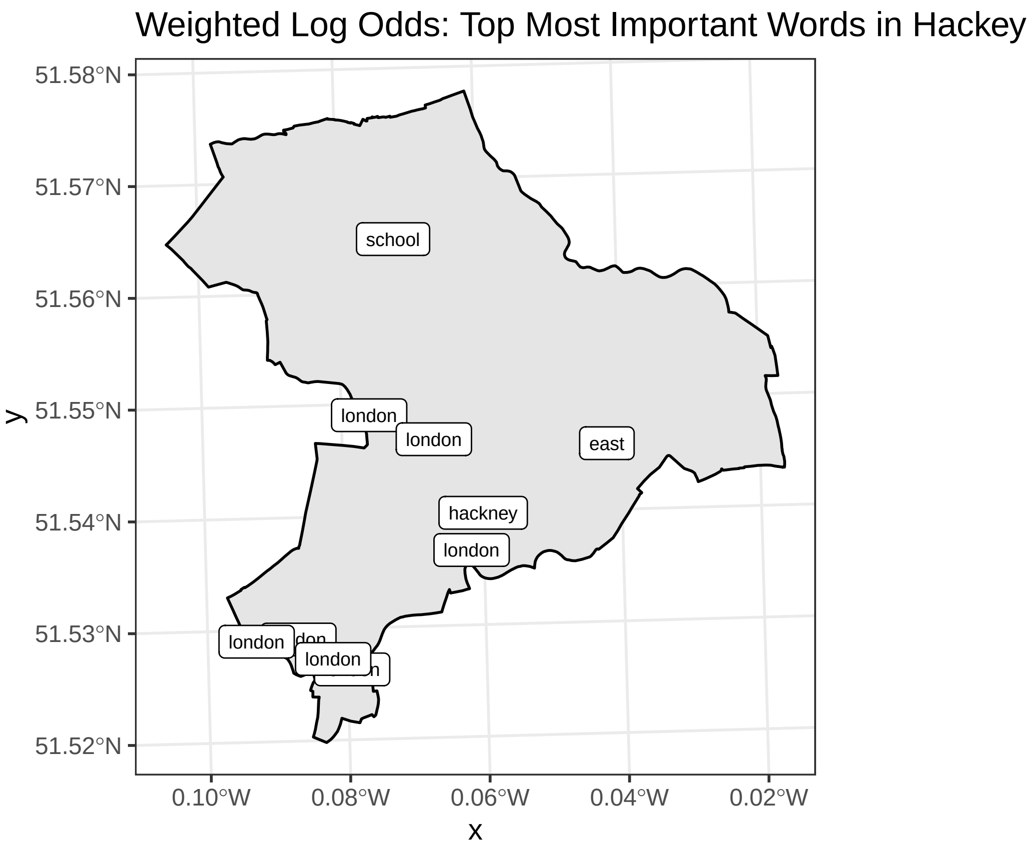
```
top10weightlog %>%
  ggplot(aes(log_odds_weighted, fct_reorder(word, log_odds_weighted),
             fill = page_title)) +
  geom_col() +
  labs(title = 'Log Weight: Top 10 Most Important Words in Hackney',
       x = 'Log Weight Score', y = 'Words') +
  theme_bw()
```




```
top10weightlogC <- top10weightlog %>% left_join(wiki_geo_coord)%>%
  st_as_sf(coords = c("gt_lon", "gt_lat"), crs = 4326) %>%
  st_transform(27700)


ggplot() +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf(data = top10weightlogC,
          aes(color = log_odds_weighted, size = log_odds_weighted )) +
  scale_size(name = "Log Weight Score")+
  scale_color_steps(name = 'Log Weight Score')+
  geom_sf_text(data = top10weightlogC, aes(label = page_title), size = 3,
               color = 'black', check_overlap = T)+
  theme_bw()+
  labs(title = 'Weighted Log Odds: Top Most Important Words in Hackey')
```




```r
ggplot(top10weightlogC) +
  geom_sf(aes(color = log_odds_weighted), show.legend = FALSE) +
  geom_sf(data = hackneyshp, color = 'black') +
  geom_sf_label(aes(label = word), size = 2.5)+
  theme_bw()+
  labs(title = 'Weighted Log Odds: Top Most Important Words in Hackey')
```



Part 3: Sentiment Analysis and Topic Modelling

Sentiment Analysis

- **Topic Modelling**: In order to understand what the collections of documents (subgroups) in a document, machine learning classification is used to classify the collection of documents into natural groups known as **"Topics"**. One of the most popular classification methods used is Latent Dirichlet allocation (LDA); it works by predicting the probability of each subgroup belonging to different topics and each topic containing a mixture of words. i.e It calculates the probability of different words being affiliated or linked to each topic, and the different topics being associated to each subgroup.
- **Sentiment Analysis**: This is another form of text classification method in Natural Language Processing. However, this already has a designated groups which text will be classified into. Sentiment analysis is often used to help companies understand people's perception about their products. It can also be regarded as a feedback analysis. It works by classifying text data into positive, negative or neutral. It can be used to identify the subjective information contained in a text.

```
#top 10 words used in all pages and their sentiment analysis
sent_pg_count <- page_word_count %>%
  inner_join(get_sentiments(lexicon = 'bing')) %>%
  count(page_title, sentiment) %>%
  slice_max(n, n=10)


#Top pages with highest positive and negative words
top10sent <- page_word_count %>%
  inner_join(get_sentiments(lexicon = 'bing')) %>%
```

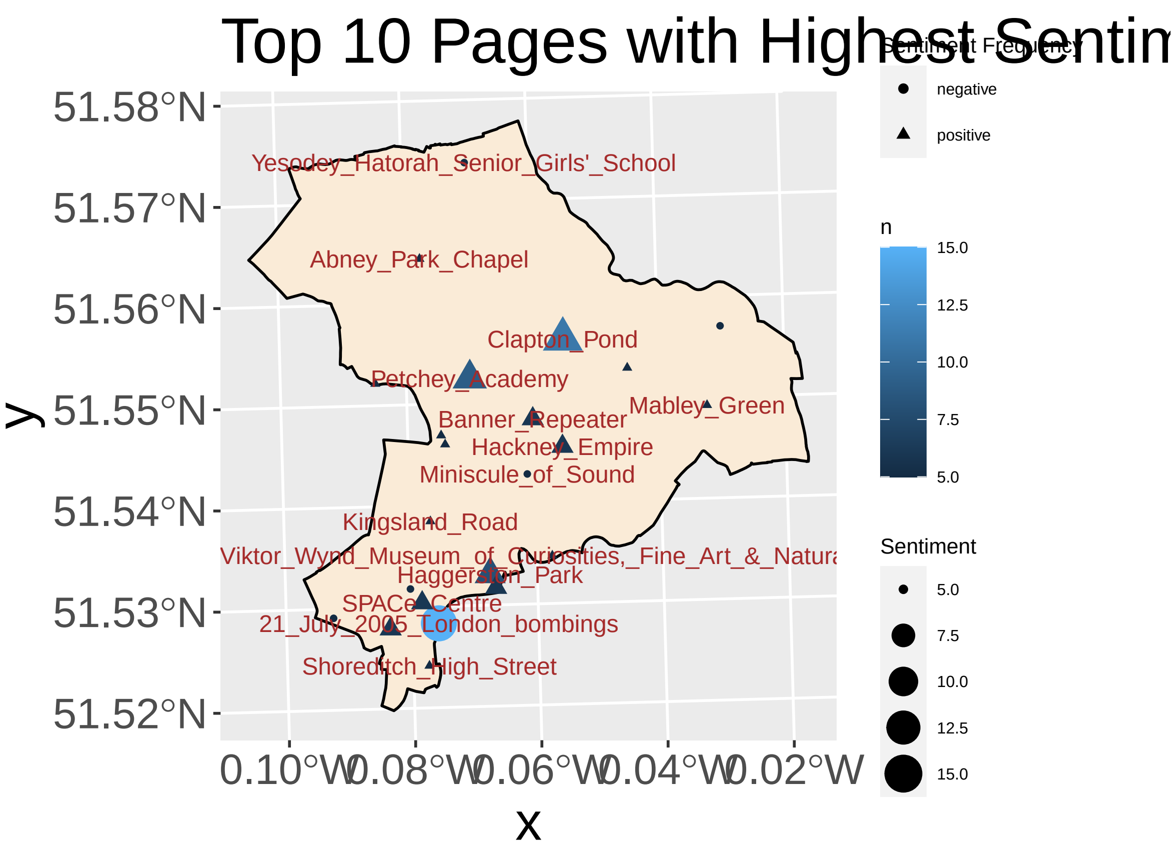
```r
  count(page_title, sentiment) %>%
  slice_max(n, n=10)

top10sent%>%
  knitr::kable(caption = "Top 10 Pages with Highest Sentiments Contributions")
```

Table 10: Top 10 Pages with Highest Sentiments Contributions

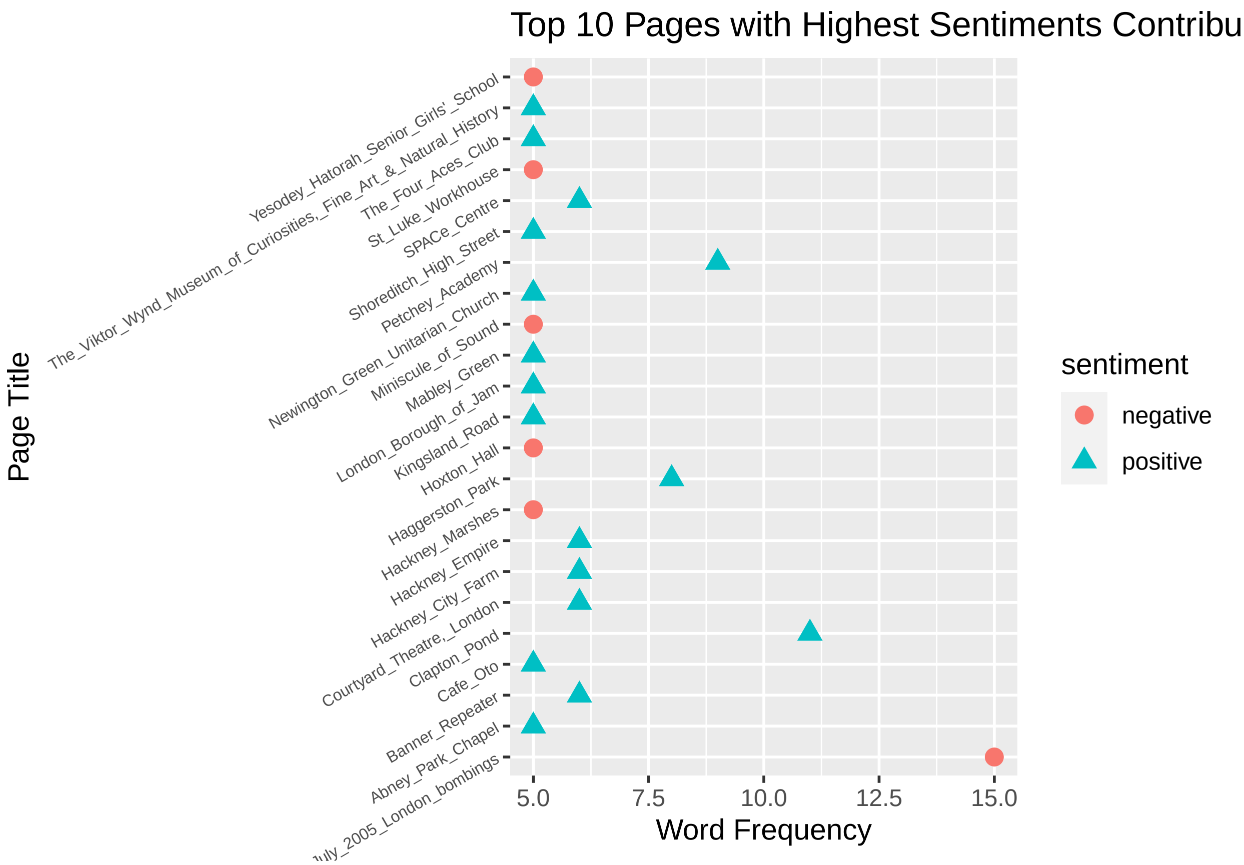
page_title	sentiment	n	geometry
21_July_2005_London_bombings	negative	15	POINT (533573.8 182801.5)
Clapton_Pond	positive	11	POINT (534936.7 185927.4)
Petchey_Academy	positive	9	POINT (533913.2 185494.1)
Haggerston_Park	positive	8	POINT (534137.9 183341.8)
Banner_Repeater	positive	6	POINT (534606.2 185050.6)
Courtyard_Theatre,_London	positive	6	POINT (533043.2 182739.3)
Hackney_City_Farm	positive	6	POINT (534203.5 183196.2)
Hackney_Empire	positive	6	POINT (534933.4 184747.7)
SPACe_Centre	positive	6	POINT (533389.7 183026.6)
Abney_Park_Chapel	positive	5	POINT (533359.7 186809.3)
Cafe_Oto	positive	5	POINT (533598 184868.2)
Hackney_Marshes	negative	5	POINT (536667 186073.9)
Hoxton_Hall	negative	5	POINT (533260.8 183179)
Kingsland_Road	positive	5	POINT (533479.9 183923.7)
London_Borough_of_Jam	positive	5	POINT (535645.8 185612.5)
Mabley_Green	positive	5	POINT (536524 185202)
Miniscule_of_Sound	negative	5	POINT (534546.5 184444.1)
Newington_Green_Unitarian_Church	positive	5	POINT (532874.1 185435.7)
Shoreditch_High_Street	positive	5	POINT (533470.3 182335.1)
St_Luke_Workhouse	negative	5	POINT (532415.5 182856.5)
The_Four_Aces_Club	positive	5	POINT (533642.3 184769.2)
The_Viktor_Wynd_Museum_of_Curiosities,*Fine_Art*&_Natural_History	positive	5	POINT (534821.6 183546.3)

page_title	sentiment	n	geometry
Yesodey_Hatorah_Senior_Girls'_School	negative	5	POINT (533851.9 187868.4)

```
ggplot() +
  geom_sf(data = hackneyshp, color = 'black', fill = "antiquewhite") +
  geom_sf(data = top10sent, aes(size = n, color = n, shape = sentiment )) +
  scale_shape(name = "Sentiment Frequency")+
  scale_size(name = 'Sentiment')+
  geom_sf_text(data = top10sent, aes(label = page_title), size = 3,
               color = 'brown', check_overlap = T)+
  labs(title = 'Top 10 Pages with Highest Sentiments Contributions', )+
  theme(legend.title = element_text(size = 8),
        legend.text  = element_text(size = 6),
        text=element_text(size=20))
```




```
#top 10 ranked pages with highest number of positive and negative words
top10sent%>%
  ggplot(aes(page_title, n, color = sentiment, shape = sentiment)) +
  geom_point(size = 3)+
  labs(title = 'Top 10 Pages with Highest Sentiments Contributions',
       x = 'Page Title', y = 'Word Frequency')+
  coord_flip()+
  theme(axis.text.y = element_text(angle = 30, hjust = 1, size = 6))
```



Page **21_July_2005_London_bombings** has the highest number of positive words Followed by **Clapton_Pond, Petchey_Academy, Haggerston_Park**. The aforementioned page titles have unique number of positive and negative words while other don't have.

```
#Top 10 positive and negative words on all Hackney pages
sentimnet_wd <- page_word_count %>%
  inner_join(get_sentiments(lexicon = 'bing'))

sum(sentimnet_wd$n)
```

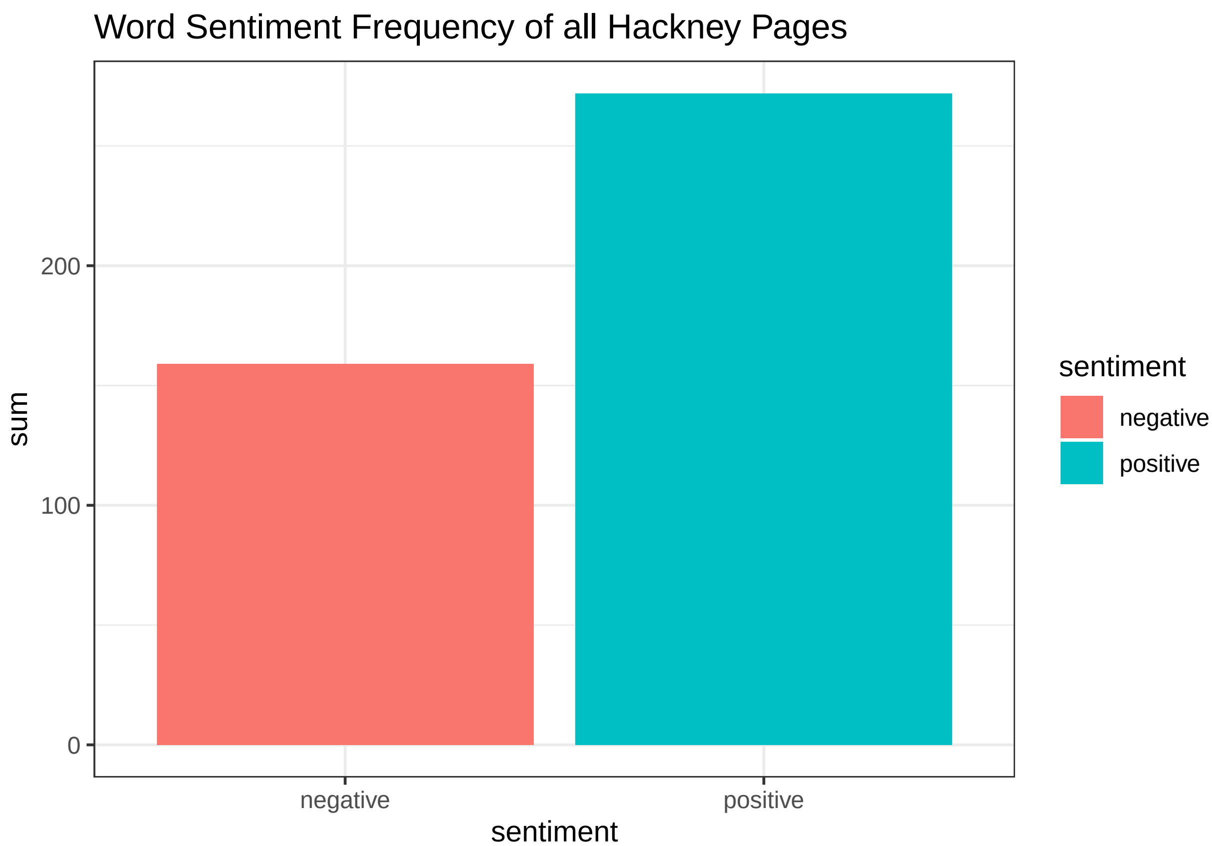
```
## [1] 483
```

```
#There are a total of 483 sentiment words in all the pages

page_word_count %>%
  inner_join(get_sentiments(lexicon = 'bing')) %>%
  count(word, sentiment) %>%
  group_by(sentiment) %>%
  summarise(sum(n)) %>%
  knitr::kable(caption = "Word Sentiment (+ve & -ve) Frequency")
```

Table 11: Word Sentiment (+ve & -ve) Frequency

sentiment	sum(n)	geometry
negative	159	MULTIPOINT ((532183.2 18321...
positive	272	MULTIPOINT ((532183.2 18321...

```
##There are 159 negative words and 272 positive words in all Hackney Pages

page_word_count %>%
  inner_join(get_sentiments(lexicon = 'bing')) %>%
  count(word, sentiment) %>%
  group_by(sentiment) %>%
  summarise(sum = sum(n))%>%
  ggplot(aes(sentiment, sum, fill= sentiment))+
  geom_col()+
  theme_bw()+
  labs(title = 'Word Sentiment Frequency of all Hackney Pages')
```



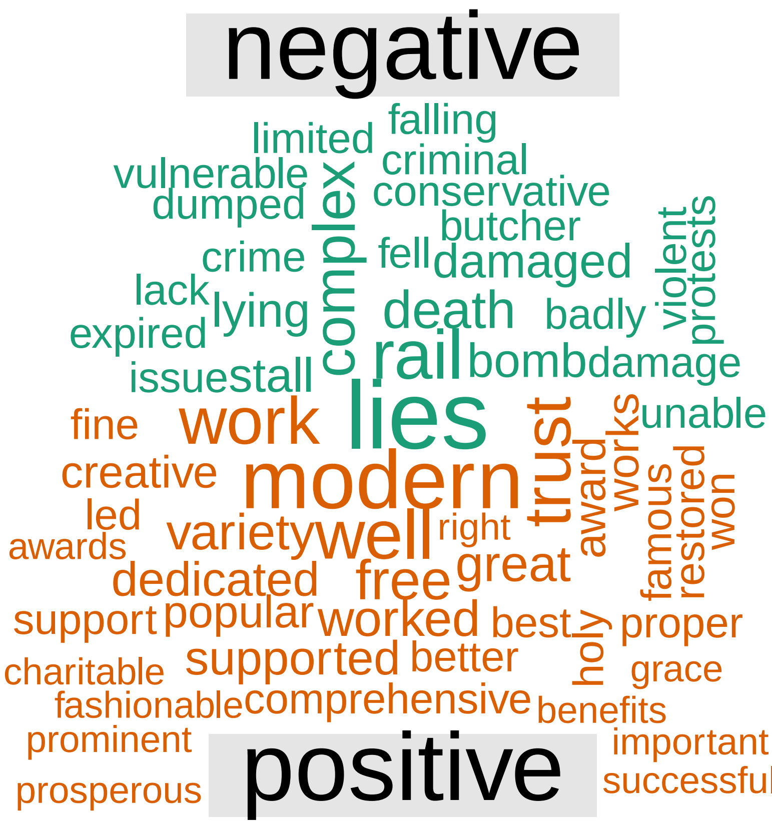
There are a total of 483 sentiment words in all the pages.
There are 159 negative words and 272 positive words in all Hackney Pages

```
#Sentiment word count
top10sent_word <- page_word_count %>%
  inner_join(get_sentiments(lexicon = 'bing')) %>%
  count(word, sentiment) %>%
  slice_max(n, n=10)

sent_word <- page_word_count %>%
  inner_join(get_sentiments(lexicon = 'bing')) %>%
  count(word, sentiment)

top10sent_word %>%
  knitr::kable(caption = "Top 10 Word Sentiment (+ve & -ve) Frequency")
```

Table 12: Top 10 Word Sentiment (+ve & -ve) Frequency

word	sentiment	n	geometry
modern	positive	16	MULTIPOINT ((532415.5 18285...
lies	negative	12	MULTIPOINT ((532183.2 18321...
well	positive	12	MULTIPOINT ((533199.9 18272...
trust	positive	11	MULTIPOINT ((532183.2 18321...
work	positive	11	MULTIPOINT ((532448.5 18750...
free	positive	8	MULTIPOINT ((532604.9 18741...
rail	negative	7	MULTIPOINT ((533483.3 18500...
great	positive	6	MULTIPOINT ((532666.8 18359...
variety	positive	6	MULTIPOINT ((532908.7 18363...
worked	positive	6	MULTIPOINT ((532238.9 18297...

```r
#Top 60 sentimental words cloud map
sent_word %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(scale=c(3, 1), max.words = 60)
```



The size of the words in the Cloud Map above is based on the frequency of the word per its sentiments. It shows us the most used positive and negative words. However, we cannot compare the size across the sentiments.

Sentiment Difference Analysis: Positive - Negative

```r
diff_sentiment <- page_word_count %>%
  inner_join(get_sentiments(lexicon = 'bing')) %>%
  count(page_title, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(diff_sent = positive - negative) %>%
  arrange(diff_sent)

#Top 10 sentiment difference
max_diff_sentiment <- diff_sentiment %>%
  slice_max(diff_sent, n=10)

max_diff_sentiment %>%
  knitr::kable(caption = "Top 10 Sentiment Difference ((+ve) - (-ve) Words) Table")
```

Table 13: Top 10 Sentiment Difference ((+ve) - (-ve) Words) Table

page_title	negative	positive	diff_sent	geometry
Clapton_Pond	2	11	9	POINT (534936.7 185927.4)
Petchey_Academy	0	9	9	POINT (533913.2 185494.1)
Hackney_Empire	0	6	6	POINT (534933.4 184747.7)
SPACe_Centre	0	6	6	POINT (533389.7 183026.6)
Banner_Repeater	1	6	5	POINT (534606.2 185050.6)
Courtyard_Theatre,_London	1	6	5	POINT (533043.2 182739.3)
Kingsland_Road	0	5	5	POINT (533479.9 183923.7)
The_Four_Aces_Club	0	5	5	POINT (533642.3 184769.2)
Abney_Park_Chapel	1	5	4	POINT (533359.7 186809.3)
Hackney_Central	0	4	4	POINT (534663.5 184496.5)
Hackney_City_Farm	2	6	4	POINT (534203.5 183196.2)
Haggerston_Park	4	8	4	POINT (534137.9 183341.8)
Hoxton_Square	0	4	4	POINT (533199.9 182721.2)
Mabley_Green	1	5	4	POINT (536524 185202)
Shoreditch_High_Street	1	5	4	POINT (533470.3 182335.1)
The_Viktor_Wynd_Museum_of_Curiosities,*Fine_Art*&_Natural_History	1	5	4	POINT (534821.6 183546.3)

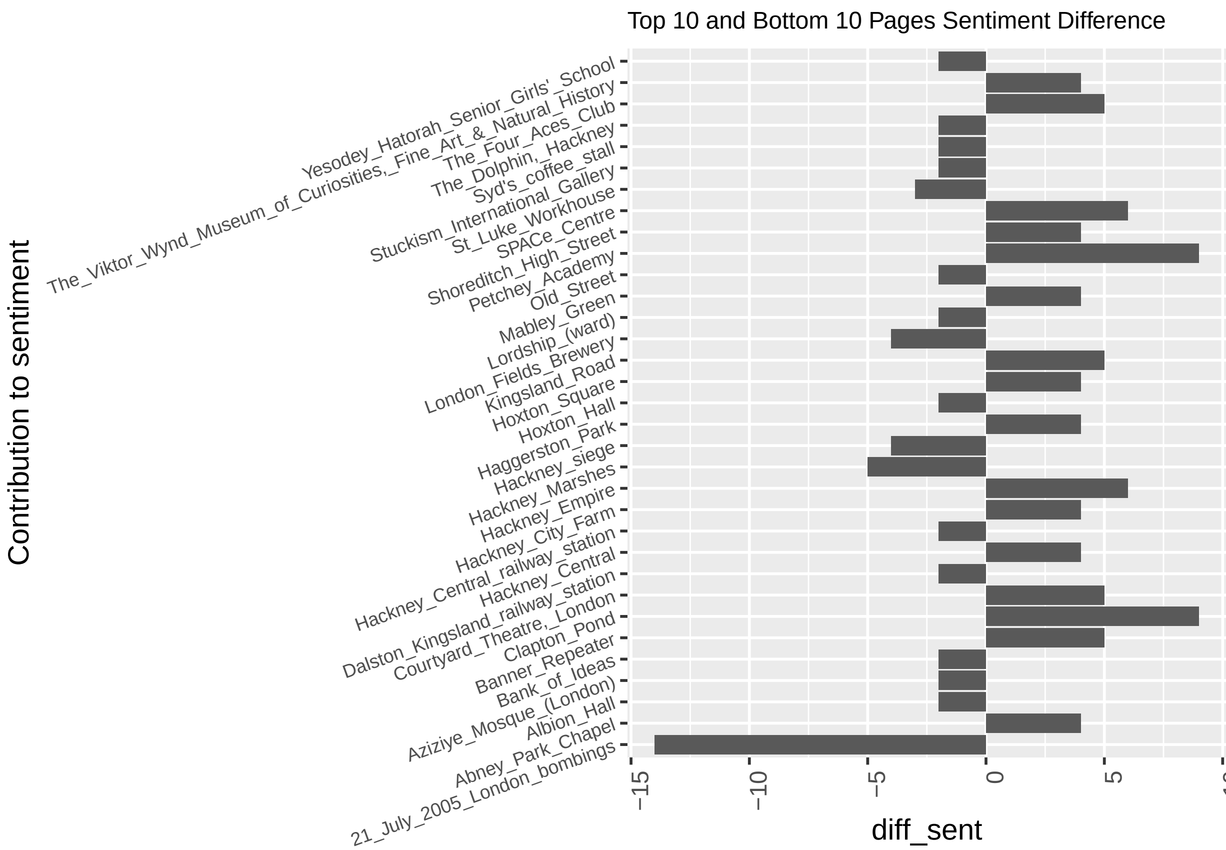
```r
#Bottom 10 sentiment difference
min_diff_sentiment <- diff_sentiment %>%
```

```r
  slice_min(diff_sent, n=10)

min_diff_sentiment %>%
  knitr::kable(caption = "Bottom 10 Sentiment Difference ((+ve) - (-ve) Words) Table")
```

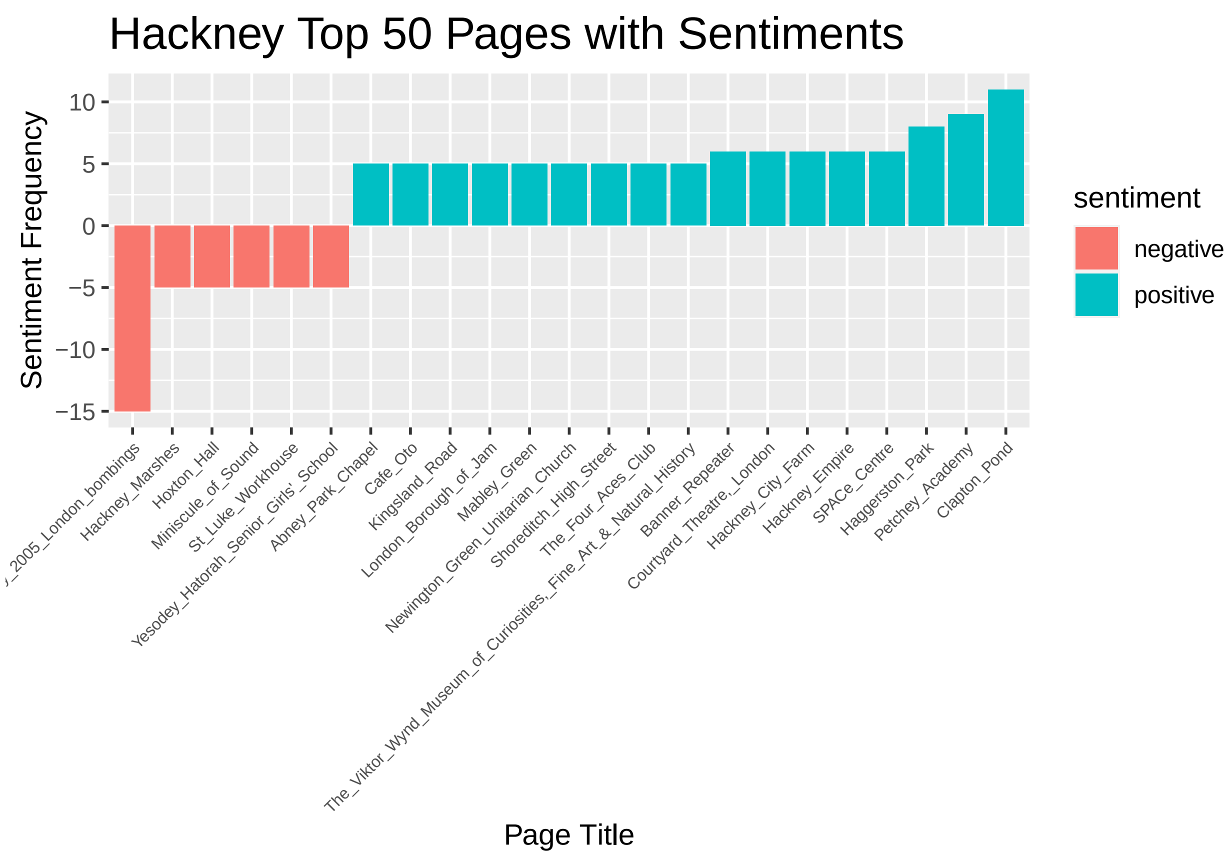
Table 14: Bottom 10 Sentiment Difference ((+ve) - (-ve) Words) Table

page_title	negative	positive	diff_sent	geometry
21_July_2005_London_bombings	15	1	-14	POINT (533573.8 182801.5)
Hackney_Marshes	5	0	-5	POINT (536667 186073.9)
Hackney_siege	4	0	-4	POINT (534694.5 184850.9)
London_Fields_Brewery	4	0	-4	POINT (534780.1 183987)
St_Luke_Workhouse	5	2	-3	POINT (532415.5 182856.5)
Albion_Hall	2	0	-2	POINT (533694.6 184072)
Aziziye_Mosque_(London)	2	0	-2	POINT (533552.7 185801.8)
Bank_of_Ideas	2	0	-2	POINT (533004.4 181882.9)
Dalston_Kingsland_railway_station	2	0	-2	POINT (533483.3 185009.9)
Hackney_Central_railway_station	2	0	-2	POINT (534901.2 184913.8)
Hoxton_Hall	5	3	-2	POINT (533260.8 183179)
Lordship_(ward)	2	0	-2	POINT (533102.6 187091.9)
Old_Street	2	0	-2	POINT (533119.4 182588.9)
Stuckism_International_Gallery	2	0	-2	POINT (533213.4 182575.6)
Syd's_coffee_stall	2	0	-2	POINT (533449.9 182583.1)
The_Dolphin,_Hackney	3	1	-2	POINT (534901.7 184112.6)
Yesodey_Hatorah_Senior_Girls'_School	5	3	-2	POINT (533851.9 187868.4)

```r
#binded rows of top 10 and bottom 10 sentiment difference
binded_sent <- bind_rows(max_diff_sentiment, min_diff_sentiment)

binded_sent %>%
  ggplot(aes(diff_sent, page_title)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        plot.title = element_text(size=9),
        axis.text.y = element_text(size=7, angle = 20)) +
  ylab("Contribution to sentiment")+
  labs(title = 'Top 10 and Bottom 10 Pages Sentiment Difference')
```




```
sent_pg_count %>%
  slice_max(n, n=50) %>%
  subset(n >= 5) %>%
  mutate(nn = ifelse(sentiment == 'positive',n, -n )) %>%
  ggplot(aes(reorder(page_title, nn), nn, fill = sentiment))+
  geom_col()+
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 6),
        plot.title = element_text(size=17)) +
  ylab("Sentiment Frequency")+
  xlab("Page Title")+
  labs(title = 'Hackney Top 50 Pages with Sentiments')
```



Topic Modelling

```
#Topic Modelling
page_tm_mt <- page_word_count %>%
  cast_dtm(page_title, word, n)


#creating 5 topics from the page topic model
latent <- LDA(page_tm_mt, k = 3, control = list(seed = 1234))

summary(latent)
```

```
##  Length   Class    Mode
##       1 LDA_VEM      S4
```

```
latent
```

```
## A LDA_VEM topic model with 3 topics.
```

```
#Word-Topic Probability
wd_tp_mdl <- tidy(latent, matrix = 'beta')

wd_tp_mdl
```

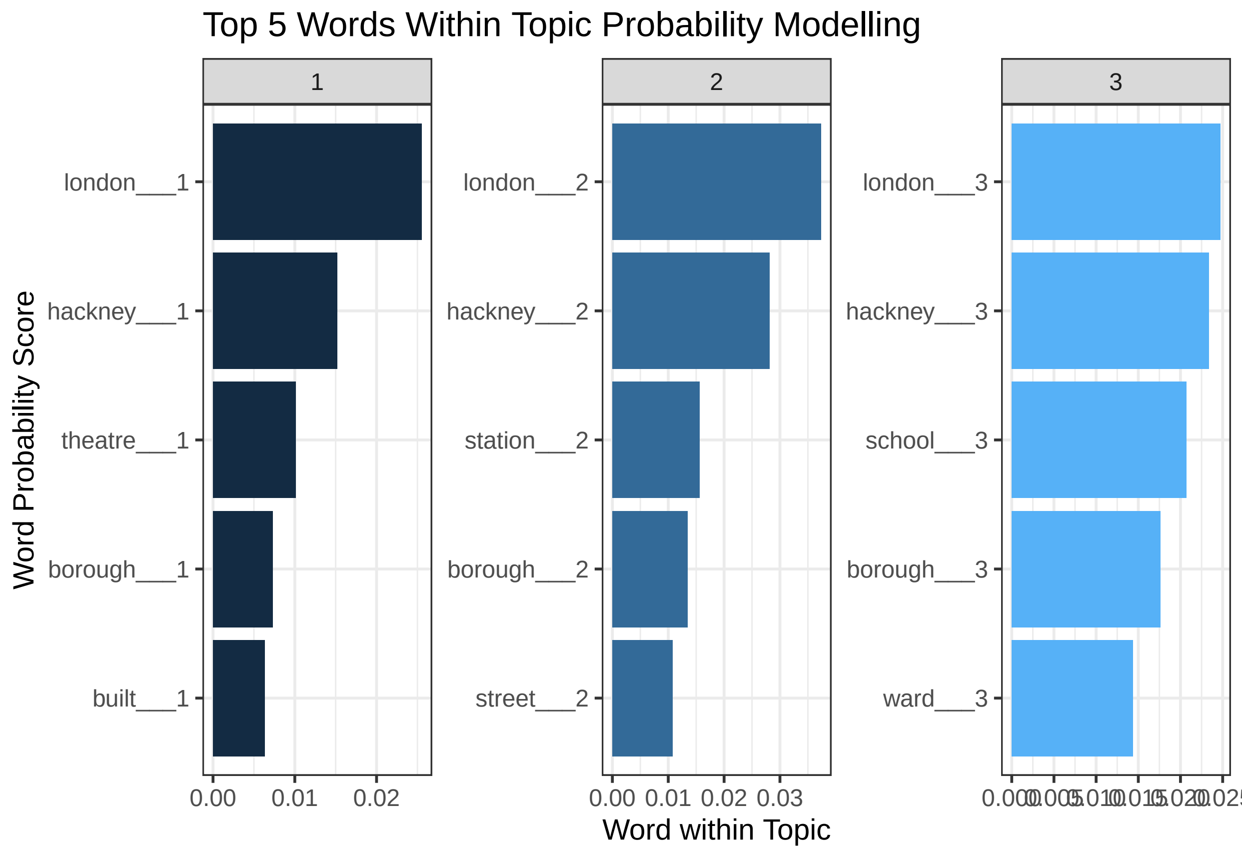
```
## # A tibble: 13,113 x 3
##    topic term          beta
##    <int> <chr>        <dbl>
##  1     1 clapton 0.00499
```

```
## 2       2 clapton 0.00232
## 3       3 clapton 0.00340
## 4       1 school  0.000000390
## 5       2 school  0.000335
## 6       3 school  0.0207
## 7       1 hackney 0.0152
## 8       2 hackney 0.0282
## 9       3 hackney 0.0234
## 10      1 road    0.00512
## # ... with 13,103 more rows
```

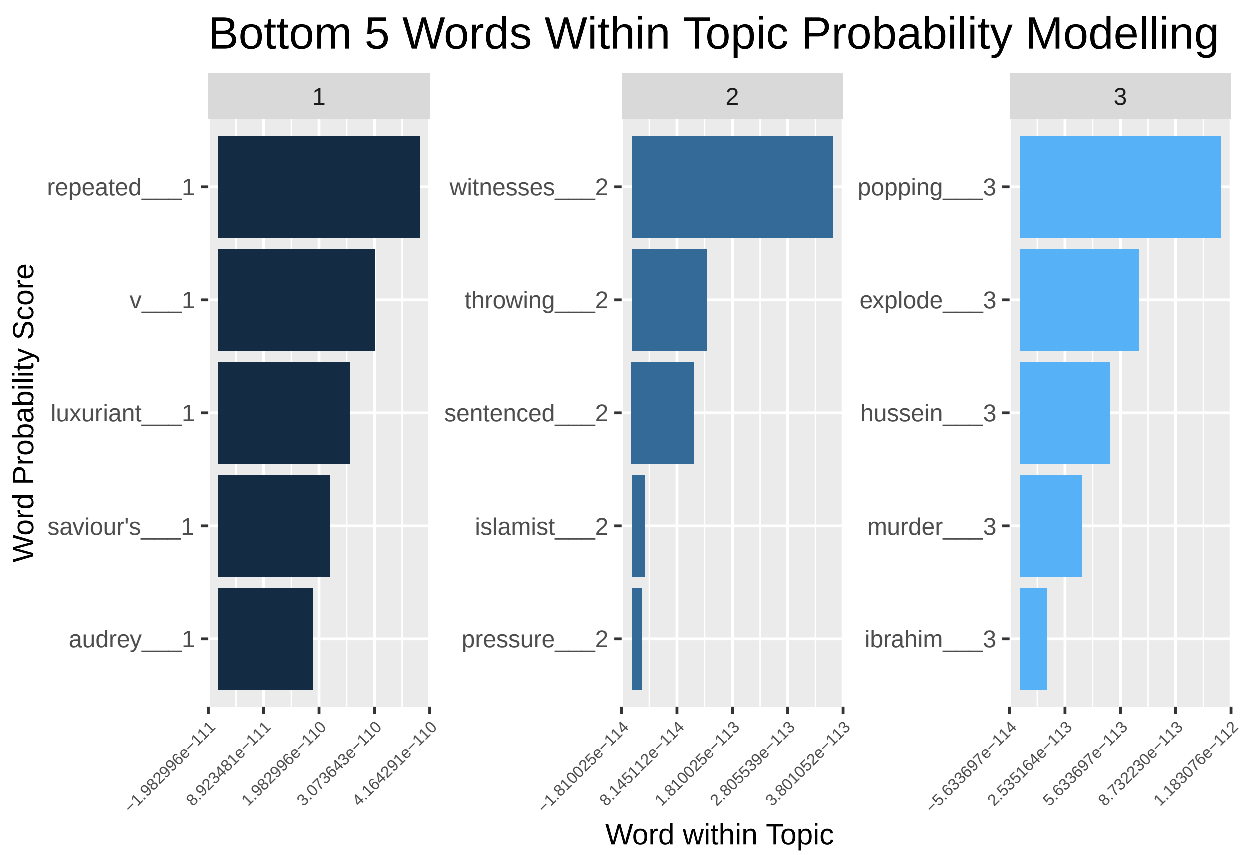
```
summary(wd_tp_mdl)
```

```
##      topic       term                 beta
##  Min.   :1   Length:13113       Min.   :0.0000000
##  1st Qu.:1   Class :character   1st Qu.:0.0000000
##  Median :2   Mode  :character   Median :0.0000000
##  Mean   :2                      Mean   :0.0002288
##  3rd Qu.:3                      3rd Qu.:0.0002438
##  Max.   :3                      Max.   :0.0373393
```

```
#Top 5 per group Word-Topic Probability
top_5_wd_tp_mdl <- wd_tp_mdl %>% group_by(topic) %>%
  slice_max(beta, n = 5) %>% ungroup()

#Bottom 5 per group Word-Topic Probability
bottom_5_wd_tp_mdl <- wd_tp_mdl %>% group_by(topic) %>%
  slice_min(beta, n = 5) %>% ungroup()

#Top 5 per group Word-Topic Probability Histogram
top_5_wd_tp_mdl %>%
  ggplot(aes(reorder_within(term, beta, topic), beta, fill = topic)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()+
  theme_bw()+
  ylab("Word within Topic")+
  xlab("Word Probability Score")+
  labs(title = 'Top 5 Words Within Topic Probability Modelling')
```




```
#Topic 1 is likely related to city centre, Topic 2: area division,
#Topic 3: Public Facilities

#Bottom 5 per group Word-Topic Probability Histogram
bottom_5_wd_tp_mdl %>%
  ggplot(aes(reorder_within(term, beta, topic), beta, fill = topic)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()+
   theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 6),
        plot.title = element_text(size=17))+
  ylab("Word within Topic")+
  xlab("Word Probability Score")+
  labs(title = 'Bottom 5 Words Within Topic Probability Modelling')
```




```
#Page_title-Topic Probability
#Word-Topic Probability
pt_tp_mdl <- tidy(latent, matrix = 'gamma')

pt_tp_mdl
```

```
## # A tibble: 723 x 3
##    document                                topic    gamma
##    <chr>                                   <int>    <dbl>
##  1 Clapton_Pond                                1 1.00
##  2 Woodberry_Down_School                       1 0.000225
##  3 Hackney_Central                             1 0.000194
##  4 Kingsland_Road                              1 0.000372
##  5 Haggerston_Park                             1 0.000137
##  6 Hackney_City_Farm                           1 0.000286
##  7 Tower_Theatre_Company                       1 1.00
##  8 Victoria_Park_railway_station_(England)     1 0.000341
##  9 21_July_2005_London_bombings                1 1.00
## 10 Church_of_St_John-at-Hackney                1 0.000273
## # ... with 713 more rows
```

- Topic 1 is likely related to city centre
- Topic 2 is likely related to area division
- Topic 3 is likely related to Public Facilities

The result shows that most of the documents were drawn from a mix of the topics. However, documents

Clapton_Pond and Church_of_St_John-at-Hackney seems to be completely drawn from topic 1.

Refrence

- GitHub. 2022. My-PGDip-Projects-/Spatial ANalysis of Close Stores.R at main · khalsz/My-PGDip-Projects-. [online] Available at: https://github.com/khalsz/My-PGDip-Projects-/blob/main/Spatial%20ANalysis%20of%20Close%20Stores.R [Accessed 21 April 2022].
- In Her Mind's Eye. 2022. My new favourite thing: weighted log odds ratios. [online] Available at: http://mindseye.sharonhoward.org/posts/my-new-favourite-thing-weighted-log-odds-ratios/ [Accessed 21 April 2022].
- Medium. 2022. TF-IDF for Document Ranking from scratch in python on real world dataset.. [online] Available at: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089 [Accessed 21 April 2022].
- Pro.arcgis.com. 2022. How Spatial Autocorrelation (Global Moran's I) works—ArcGIS Pro | Documentation. [online] Available at: https://pro.arcgis.com/en/pro-app/2.8/tool-reference/spatial-statistics/h-how-spatial-autocorrelation-moran-s-i-spatial-st.htm#:~:text=The%20Spatial%20Autocorrelation%20(Global%20Moran's,clustered%2C%20dispersed%2C%20or%20random. [Accessed 21 April 2022].
- Robinson, J., 2022. 6 Topic modeling | Text Mining with R. [online] Tidytextmining.com. Available at: https://www.tidytextmining.com/topicmodeling.html [Accessed 21 April 2022].

- Robinson, J., 2022. 6 Topic modeling | Text Mining with R. [online] Tidytextmining.com. Available at: https://www.tidytextmining.com/topicmodeling.html [Accessed 21 April 2022].