Week 9: Web APIs

Alina Vikhnevich

2025-04-13

Introduction

For this assignment, I explored how to connect to a RESTful API and extract structured data from the web. I used the *New York Times Most Popular API*, which provides access to a feed of the most-emailed articles over the past week. The overall objective was to pull this data in JSON format, parse it, clean it, and finally transform it into a tidy R `data.frame`, which I then saved as an Excel file for future use.

This exercise gave me hands-on experience with API calls using `httr2`, working with JSON responses, and cleaning nested data structures skills that are crucial for web-based data acquisition and preparation in real-world data science workflows.

Load Required Libraries

```
# Load required libraries
library(httr2)
library(jsonlite)
library(tidyverse)
library(tidytext)
library(writexl)
```

I started by loading all the libraries needed to complete the task. `httr2` is used to perform the HTTP request to the API. `jsonlite` helps parse the JSON response into a format that R can work with. `tidyverse` is essential for data wrangling, and `writexl` lets me export the cleaned dataset to Excel for easier review and sharing.

1. Store API Key and Send the Request

```
# Store your API key
api_key <- Sys.getenv("NYT_API_KEY")

# Create and send the request
resp <- request("https://api.nytimes.com/svc/mostpopular/v2/emailed/7.json") %>%
  req_url_query("api-key" = api_key) %>%
  req_perform()
```

I registered for an API key from the NYT Developer Portal. To securely manage access credentials, I stored my New York Times API key in the `.Renviron` file and retrieved it in the script using

`Sys.getenv("NYT_API_KEY")`. It keeps the key hidden from the rendered document. Then, I constructed the API request using `httr2`, added the key as a query parameter, and performed the request. This returned a live HTTP response containing JSON data of the top-emailed articles.

2. Parse JSON Content

```
# Parse the JSON content
resp_text <- resp_body_string(resp)
data_parsed <- fromJSON(resp_text, flatten = TRUE)
```

The raw JSON content from the API response was first converted into a character string, and then parsed using `fromJSON()`. I used `flatten = TRUE` so that any nested data structures were simplified into a flat data frame format. This helps avoid complex list-columns later in the analysis.

3. Extract the Articles Section

```
# Extract just the 'results' section
articles_df <- data_parsed$results

# Take a quick look
glimpse(articles_df)
```

```
## Rows: 20
## Columns: 22
## $ uri            <chr> "nyt://article/24d1c55e-c9c6-5511-ae14-4a2ce47b9e0b", "~
## $ url            <chr> "https://www.nytimes.com/2025/04/07/business/china-manu~
## $ id             <dbl> 1e+14, 1e+14, 1e+14, 1e+14, 1e+14, 1e+14, 1e+14, 1e+14,~
## $ asset_id       <dbl> 1e+14, 1e+14, 1e+14, 1e+14, 1e+14, 1e+14, 1e+14, 1e+14,~
## $ source         <chr> "New York Times", "New York Times", "New York Times", "~
## $ published_date <chr> "2025-04-07", "2025-04-10", "2025-04-09", "2025-04-10",~
## $ updated        <chr> "2025-04-08 11:25:15", "2025-04-10 19:35:19", "2025-04-~
## $ section        <chr> "Business", "Opinion", "Opinion", "Well", "Opinion", "T~
## $ subsection     <chr> "", "", "", "", "", "", "", "", "Move", "", "", "", "Bo~
## $ nytdsection    <chr> "business", "opinion", "opinion", "well", "opinion", "t~
## $ adx_keywords   <chr> "International Trade and World Market;Factories and Man~
## $ column         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ byline         <chr> "By Keith Bradsher", "By John McWhorter", "By Jamelle B~
## $ type           <chr> "Article", "Article", "Article", "Article", "Article", ~
## $ title          <chr> "'The Tsunami Is Coming': China's Global Exports Are Ju~
## $ abstract       <chr> "A staggering $1.9 trillion in extra industrial lending~
## $ des_facet      <list> <"International Trade and World Market", "Factories an~
## $ org_facet      <list> <>, "Metro-Goldwyn-Mayer Inc", "Republican Party", <>,~
## $ per_facet      <list> "Trump, Donald J", <"Astaire, Fred", "Bergman, Ingrid"~
## $ geo_facet      <list> "China", <>, <>, <>, <>, <"Carmel (Calif)", "Carmel Va~
## $ media          <list> [<data.frame[1 x 6]>], [<data.frame[1 x 6]>], [<data.f~
## $ eta_id         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
```

The API response includes metadata along with the actual content we're interested in. I isolated the `results` list, which contains the articles themselves, and assigned it to a new data frame.

4. Clean and Tidy the Dataset

```r
# Clean and structure the data
clean_articles <- articles_df %>%
  select(
    title,
    byline,
    section,
    published_date,
    source,
    abstract,
    url
  ) %>%
  mutate(across(everything(), ~ ifelse(. == "" | is.na(.), "None", .)))

# Preview the cleaned dataset
head(clean_articles)
```

```
##                                                                    title
## 1 'The Tsunami Is Coming': China's Global Exports Are Just Getting Started
## 2                       Why These 10 Old Movies Are Really Worth Your Time
## 3                                    The Tariff Saga Is About One Thing
## 4            5 Science-Backed Longevity 'Hacks' That Don't Cost a Fortune
## 5                  Why Did So Many People Delude Themselves About Trump?
## 6                                          36 Hours in Carmel, Calif.
##                   byline  section published_date         source
## 1      By Keith Bradsher Business     2025-04-07 New York Times
## 2      By John McWhorter  Opinion     2025-04-10 New York Times
## 3       By Jamelle Bouie  Opinion     2025-04-09 New York Times
## 4 By Mohana Ravindranath     Well     2025-04-10 New York Times
## 5   By Michelle Goldberg  Opinion     2025-04-07 New York Times
## 6    By DANIEL SCHEFFLER   Travel     2025-04-03 New York Times
##
## 1 A staggering $1.9 trillion in extra industrial lending is fueling a continued flood of exports tha
## 2
## 3                                                                                          The tariff s
## 4                                                                                               You do
## 5
## 6                                On California's Central Coast, three storybook enclaves draw visito
##                                                                                                   url
## 1 https://www.nytimes.com/2025/04/07/business/china-manufacturing-exports-trump-tariffs.html
## 2              https://www.nytimes.com/2025/04/10/opinion/movies-technology-old-america.html
## 3              https://www.nytimes.com/2025/04/09/opinion/trump-tariffs-rationale-power.html
## 4                       https://www.nytimes.com/2025/04/10/well/longevity-low-cost-tips.html
## 5             https://www.nytimes.com/2025/04/07/opinion/trump-stock-market-wall-street.html
## 6             https://www.nytimes.com/interactive/2025/04/03/travel/things-to-do-carmel.html
```

The raw dataset contained many columns, some of which were either nested or not useful for my purposes. I kept only the key columns needed for a readable summary: title, author, section, publication date, source, summary, and URL. To handle missing `byline` entries, I replaced empty or `NA` values with `"No Author"`, which keeps the dataset more readable.

5. Analysis

1. Publishing Trends Over Time

To begin understanding trends in reader engagement, I first looked at how many top-emailed articles were published each day. I converted the `published_date` column to a proper `Date` format so I could group and count articles by date. Then, I visualized the frequency of publications over time.

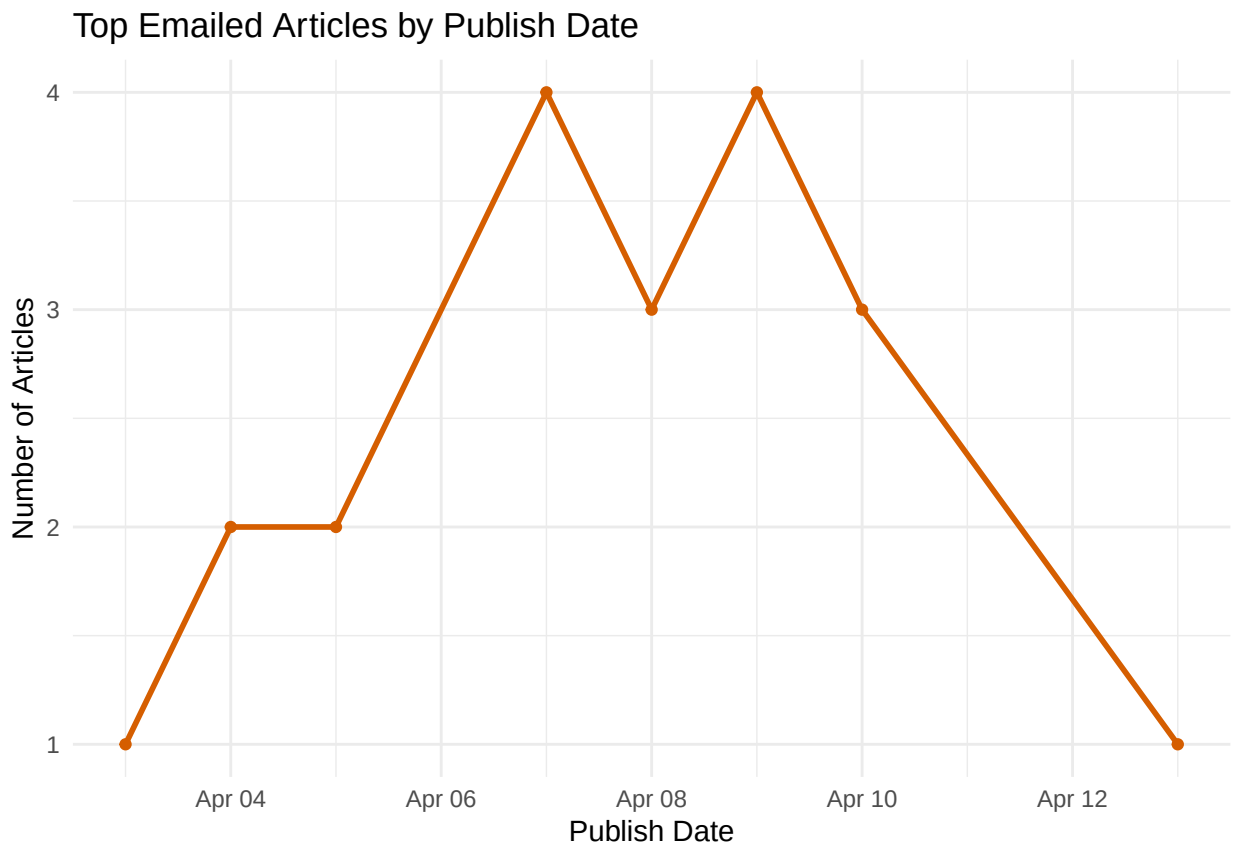
This helps reveal whether publication volume or timing might influence popularity. For instance, if articles published mid-week are more likely to trend, that insight could guide future content planning.

```
# Convert published_date to Date type
clean_articles <- clean_articles %>%
  mutate(published_date = as.Date(published_date))

# Count articles by date
date_counts <- clean_articles %>%
  group_by(published_date) %>%
  summarise(count = n())

# Plot
ggplot(date_counts, aes(x = published_date, y = count)) +
  geom_line(size = 1, color = "#D55E00") +
  geom_point(color = "#D55E00") +
  labs(
    title = "Top Emailed Articles by Publish Date",
    x = "Publish Date",
    y = "Number of Articles"
  ) +
  theme_minimal()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



2. Who Are the Most Frequent Authors?

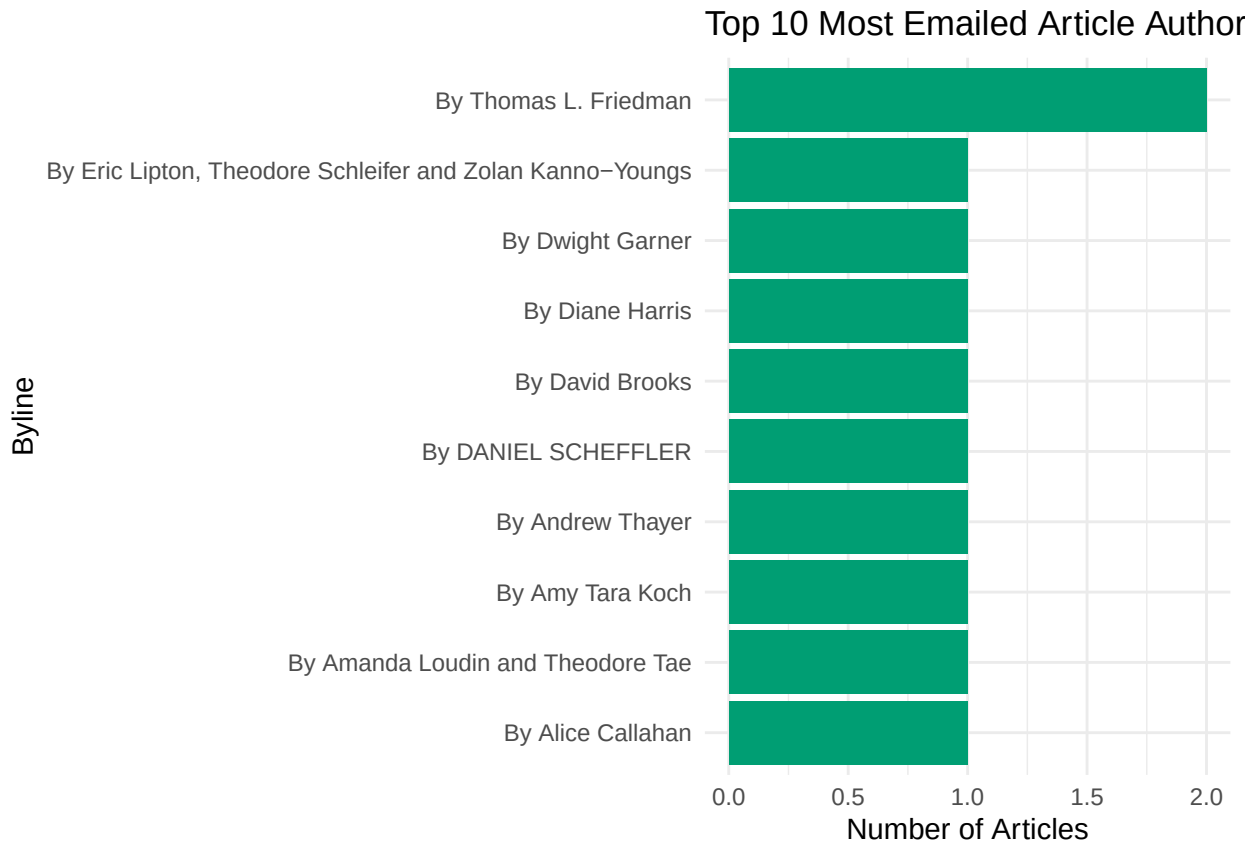
Next, I wanted to know who's behind the most popular content. By grouping articles by the `byline` field and filtering out missing authors, I counted how many top-emailed articles each author had.

The visualization highlights the top 10 most frequently featured authors, which can help uncover whose writing consistently resonates with readers. It also gives insight into the kind of voices NYT audiences prefer.

```
# Count non-empty bylines
author_counts <- clean_articles %>%
  filter(byline != "No Author") %>%
  count(byline, sort = TRUE)

# Display top 10
top_authors <- head(author_counts, 10)

# Bar chart
ggplot(top_authors, aes(x = reorder(byline, n), y = n)) +
  geom_col(fill = "#009E73") +
  labs(
    title = "Top 10 Most Emailed Article Authors",
    x = "Byline",
    y = "Number of Articles"
  ) +
  coord_flip() +
  theme_minimal()
```



3. Text Analysis: Word Frequency in Titles

I thought it would be interesting to explore which words appear most often in article titles, as titles often drive engagement. After loading common stop words to filter out generic language like "the" or "and", I tokenized the title text into individual words.

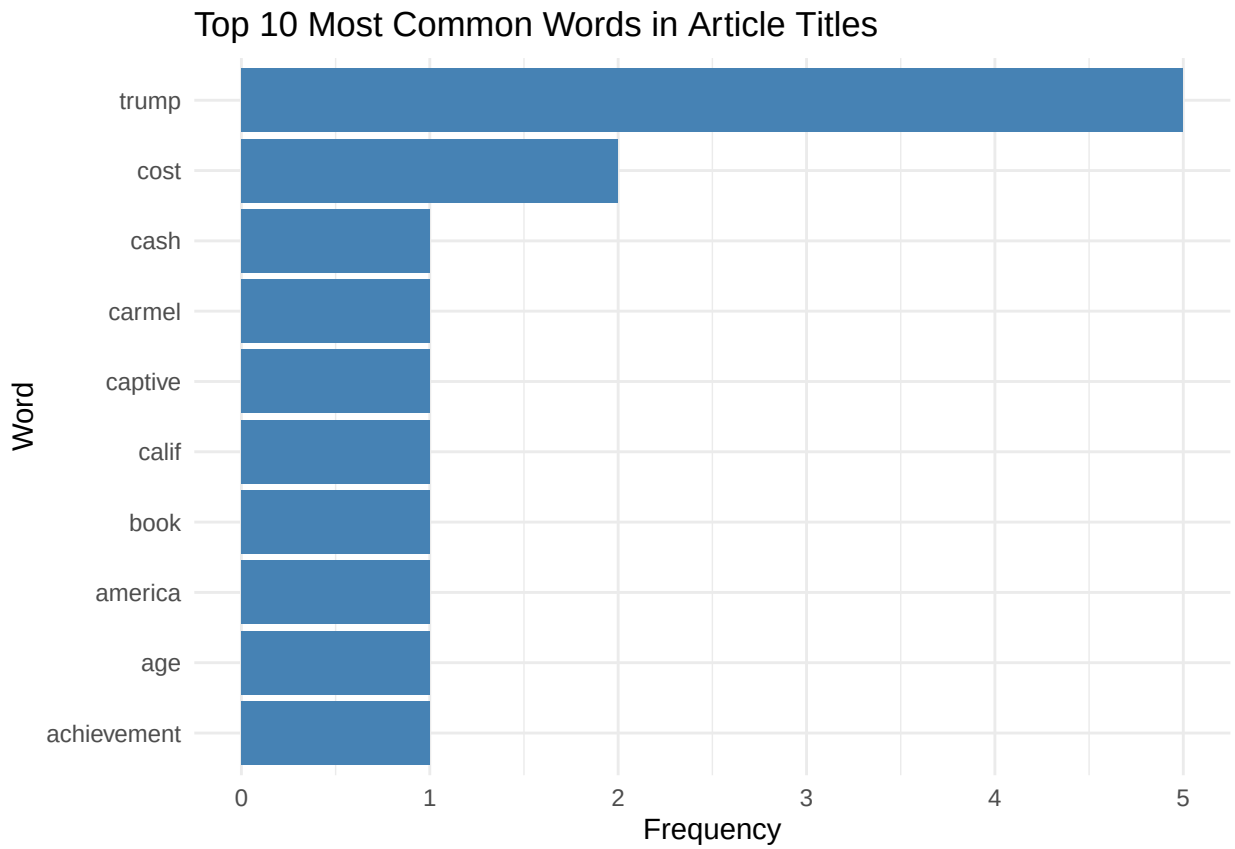
I then counted and ranked these words, removing pure numeric tokens. This shows the recurring themes or entities (like "Trump" in this dataset) that frequently appear in widely shared articles.

```
data("stop_words")

# Clean and tokenize
title_words <- clean_articles %>%
  unnest_tokens(word, title) %>%
  filter(!word %in% stop_words$word) %>%
  filter(!str_detect(word, "^\\d+$")) %>%  # Remove pure numbers
  count(word, sort = TRUE) %>%
  arrange(desc(n), word) %>%
  head(10)

# Plot
ggplot(title_words, aes(x = reorder(word, n), y = n)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(
    title = "Top 10 Most Common Words in Article Titles",
    x = "Word",
```

```
    y = "Frequency"
  ) +
  theme_minimal()
```



4. Frequency Analysis

Lastly, I looked at which news sections are most commonly associated with top-emailed articles. Grouping by the `section` field and counting the occurrences gave a clear picture of what types of content NYT readers are most likely to share.

Not surprisingly, the Opinion section dominated, which suggests that readers are often drawn to analysis, perspectives, and editorial pieces. This insight helps in understanding content type popularity across the site.

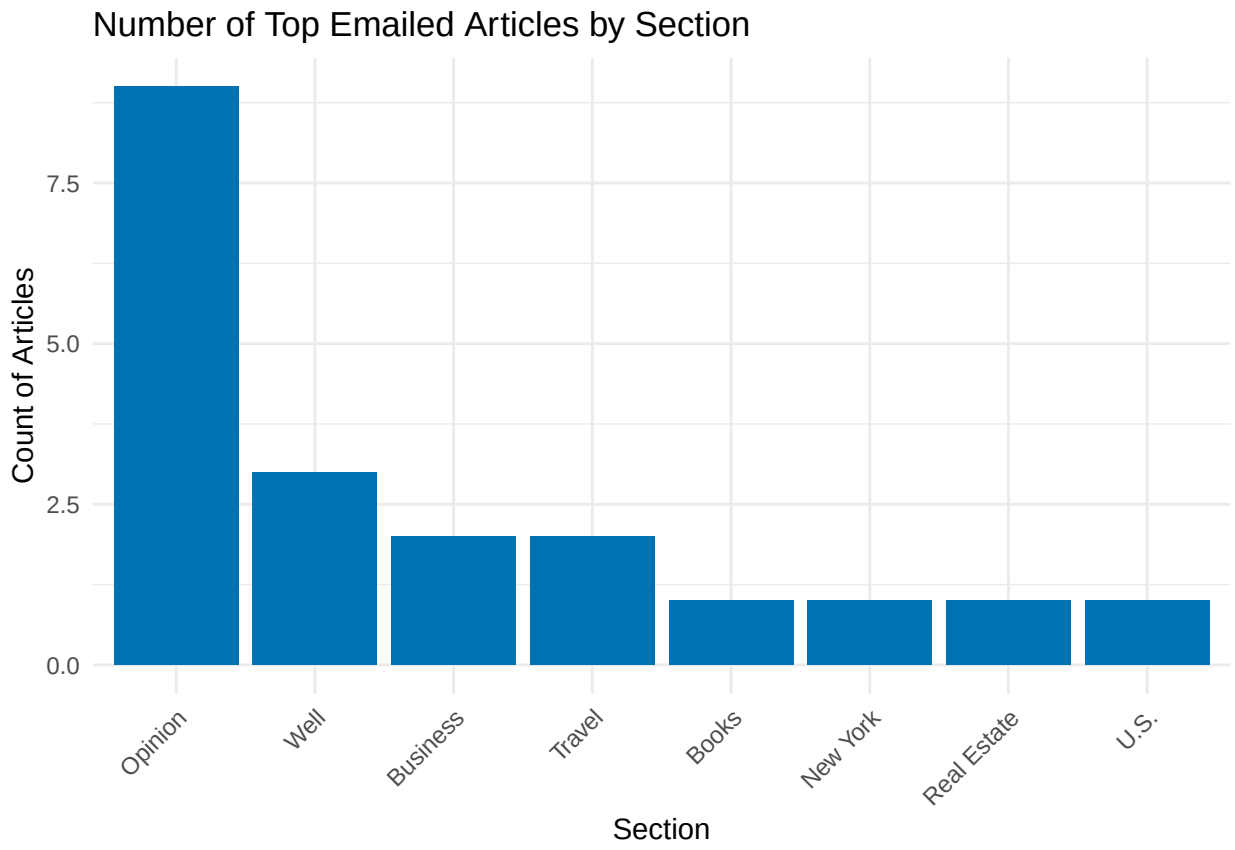
```
# Count number of articles per section
section_summary <- clean_articles %>%
  group_by(section) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

# Show summary table
print(section_summary)
```

```
## # A tibble: 8 x 2
##   section     count
##   <chr>       <int>
```

```
## 1 Opinion         9
## 2 Well            3
## 3 Business        2
## 4 Travel          2
## 5 Books           1
## 6 New York        1
## 7 Real Estate     1
## 8 U.S.            1
```

```r
# Visualize it
ggplot(section_summary, aes(x = reorder(section, -count), y = count)) +
  geom_bar(stat = "identity", fill = "#0072B2") +
  labs(title = "Number of Top Emailed Articles by Section",
       x = "Section",
       y = "Count of Articles") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



6. Save the Results to Excel

```r
# Save cleaned version to Excel file
write_xlsx(clean_articles, "nyt_popular_articles.xlsx")
```

Once the data was cleaned, I exported it to an Excel file. I chose .xlsx instead of .csv to avoid encoding issues with special characters (e.g., apostrophes and quotation marks) that showed up when exporting to CSV. This method ensured all article titles and abstracts remained readable and intact.

Conclusion

This assignment provided a full-cycle experience of working with a web API from authentication and data acquisition to cleaning, transformation, and exploratory analysis. Using the `httr2` package, I authenticated my request with a securely stored API key, then retrieved live JSON data from the New York Times Most Popular API.

After parsing the JSON with `jsonlite::fromJSON()` and flattening the nested structure, I filtered and cleaned the dataset to keep only the most relevant information such as article title, author, section, and abstract. Missing author names were handled with fallback values to ensure readability.

The analysis phase offered insight into publishing patterns and user engagement. I explored how article popularity related to publish date, most frequent authors, word choice in titles, and section categories. Notably, the *Opinion* section appeared most often, and terms like "Trump" frequently surfaced in popular article titles highlighting trends in what captures readers' attention.

A small challenge arose with character encoding when exporting to CSV, which was resolved by switching to `.xlsx` using `writexl::write_xlsx()`.

This project reinforced my ability to work with APIs and transform semi-structured data into actionable insights using R a critical skill for real-world data science applications.