Web Scraping using R: Extracting and Analyzing Journal Article Data

Journal: Translational Neurodegeneration

Ethalapaka Surya Teja, Shiva Karthik Pinjarle Manmohan, Ibrahim Musa

Introduction and objectives:

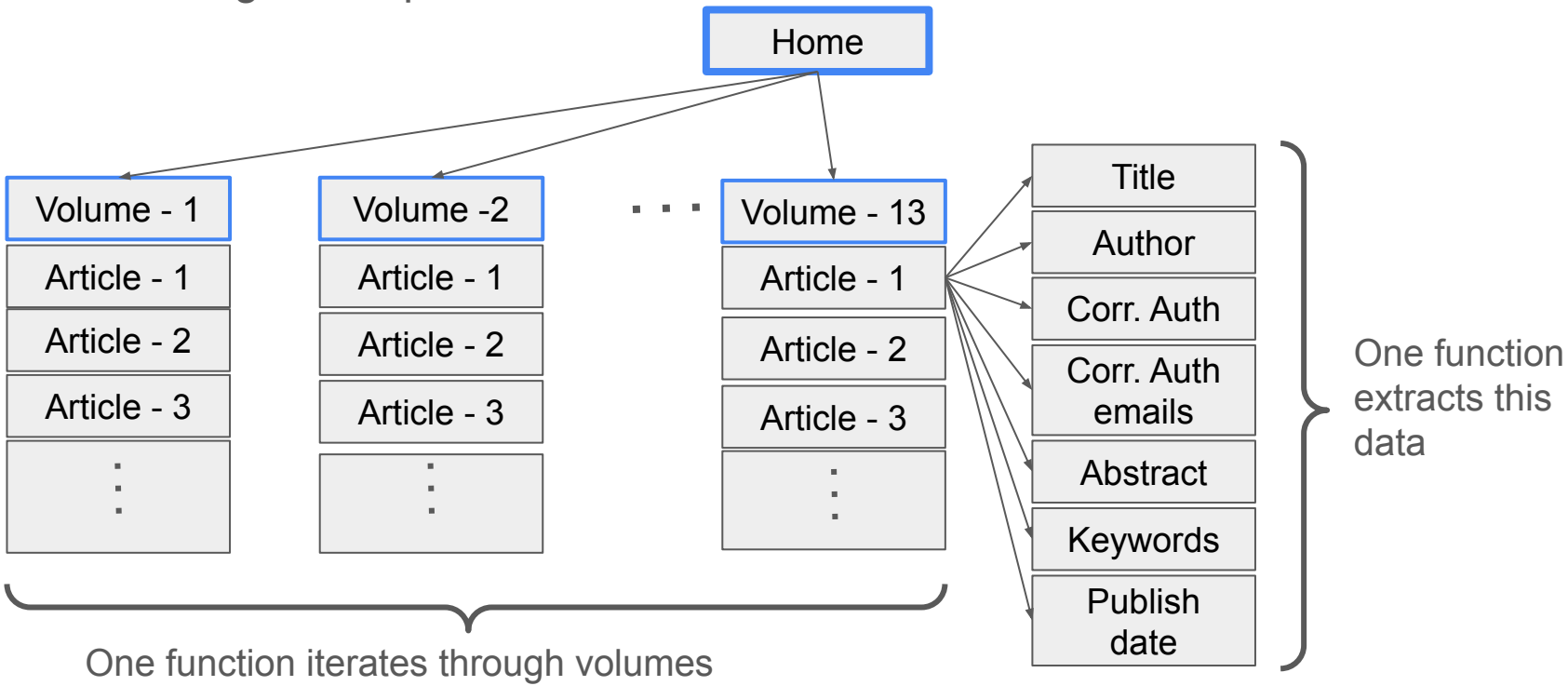
Main objective: To build a specialized R program to crawl, parse, and extract useful information from your selected journal.

Steps:

1. Scraping
2. Cleaning and Pre-processing
3. Analysing and Visualization

Scraping : Idea

Understanding sitemap:



Scraping code(extracting):

```r
extract_article_info <- function(url) {
  webpage <- read_html(url)
  Title <- webpage %>% html_nodes("#main-content > main > article > div.c-article-header > h1") %>% html_text()
  authxpath <- "#Aff1 > p.c-article-author-affiliation__authors-list"
  authors <- webpage %>% html_nodes(authxpath) %>% html_text()
  author_paragraph <- webpage %>% html_node("#corresponding-author-list")

  if (!is.null(author_paragraph)) {
    author_links <- author_paragraph %>% html_nodes("a")
    author_names <- html_text(author_links)
    author_emails <- author_links %>% html_attr("href")
    author_emails <- gsub("mailto:", "", author_emails)
    corresp_authors <- author_names
    corresp_emails <- author_emails
  } else {
    corresp_authors <- NA
    corresp_emails <- NA
  }

  publish_Date <- webpage %>% html_nodes("#main-content > main > article > div.c-article-header > ul.c-article-identifiers > li:nth-child(3) > a >
  Abstract <- webpage %>% html_nodes("#Abs1-content > p") %>% html_text()
  css_selector_keywords <- "#article-info-content > div > div > ul.c-article-subject-list"
  keywords_element <- webpage %>% html_node(css_selector_keywords)
  keywords_vector <- character(0)

  if (!is.null(keywords_element)) {
    keyword_spans <- keywords_element %>% html_nodes("span")
    keywords_vector <- html_text(keyword_spans)
  }

  return(list(
    Title = Title,
    Authors = authors,
    CorrespondingAuthors = paste(corresp_authors, collapse = ", "),
    CorrespondingAuthorsEmails = paste(corresp_emails, collapse =", "),
    PublishDate = publish_Date,
    Abstract = Abstract,
    Keywords = paste(keywords_vector, collapse = ", ")
  ))
```

Scraping code(iterating):

```r
scrape_articles_by_volume <- function(volume) {
  base_url <- "https://translationalneurodegeneration.biomedcentral.com"
  article_info_list <- list()
  url <- paste0(base_url, "/articles?query=&volume=", volume, "&searchType=&tab=keyword")
  webpage <- read_html(url)
  has_pagination <- length(webpage %>% html_nodes("#main-content > div > main > div:nth-child(3) > nav > ul > li")) > 3

  if (has_pagination) {
    total_pages <- length(webpage %>% html_nodes("#main-content > div > main > div:nth-child(3) > nav > ul > li") %>% html_text()) - 2
  } else {
    total_pages <- 1
  }

  for (page in 1:total_pages) {
    url <- paste0(base_url, "/articles?query=&volume=", volume, "&searchType=&tab=keyword&page=",page)
    webpage <- read_html(url)
    article_links <- webpage %>% html_nodes("#main-content > div > main > div:nth-child(3) > ol > li article h3 a") %>% html_attr("href")
    article_info <- lapply(article_links, function(article_link) {
      full_article_url <- paste0(base_url, article_link)
      return(extract_article_info(full_article_url))
    })
    article_info_list <- c(article_info_list, article_info)
  }

  return(article_info_list)
}
```

Cleaning:

```r
# Check for duplicates based on Title
duplicate_titles <- duplicated(articles_df$Title)

# Identify rows to keep (first occurrences of duplicates)
rows_to_keep <- !duplicate_titles

# Filter the dataframe to keep rows where Title is not a duplicate
nonduplicate_articles_df <- articles_df[rows_to_keep, ]
```

Remove duplicates by keeping only first occurrences

```r
# Convert string "NA" to actual NA (missing value)
nonduplicate_articles_df[nonduplicate_articles_df == "NA"] <- NA

# Check for NA values
any_na <- any(is.na(nonduplicate_articles_df))

if (any_na) {
  # If NA values exist, identify which columns contain NA values
  na_columns <- colnames(nonduplicate_articles_df)[colSums(is.na(nonduplicate_articles_df)) > 0]

  # Display the columns with NA values
  print("Columns with NA values:")
  print(na_columns)

  # Show rows and columns with NA values
  print("Rows and columns with NA values:")
  print(which(is.na(nonduplicate_articles_df), arr.ind = TRUE))
} else {
  print("No NA values in the dataframe.")
}

# Remove rows with NA values
cleaned_project_dataframe <- nonduplicate_articles_df[complete.cases(nonduplicate_articles_df), ]
```

Checking if NA values are present,
And if present, removing them.

Preprocessing:

```r
# Convert 'Publish_Date' to 'YYYY-MM-DD' format
cleaned_project_dataframe$PublishDate <- as.Date(cleaned_project_dataframe$PublishDate, format = "%d %B %Y")

cleaned_project_dataframe$PublishDate <- format(cleaned_project_dataframe$PublishDate, "%Y-%m-%d")
```

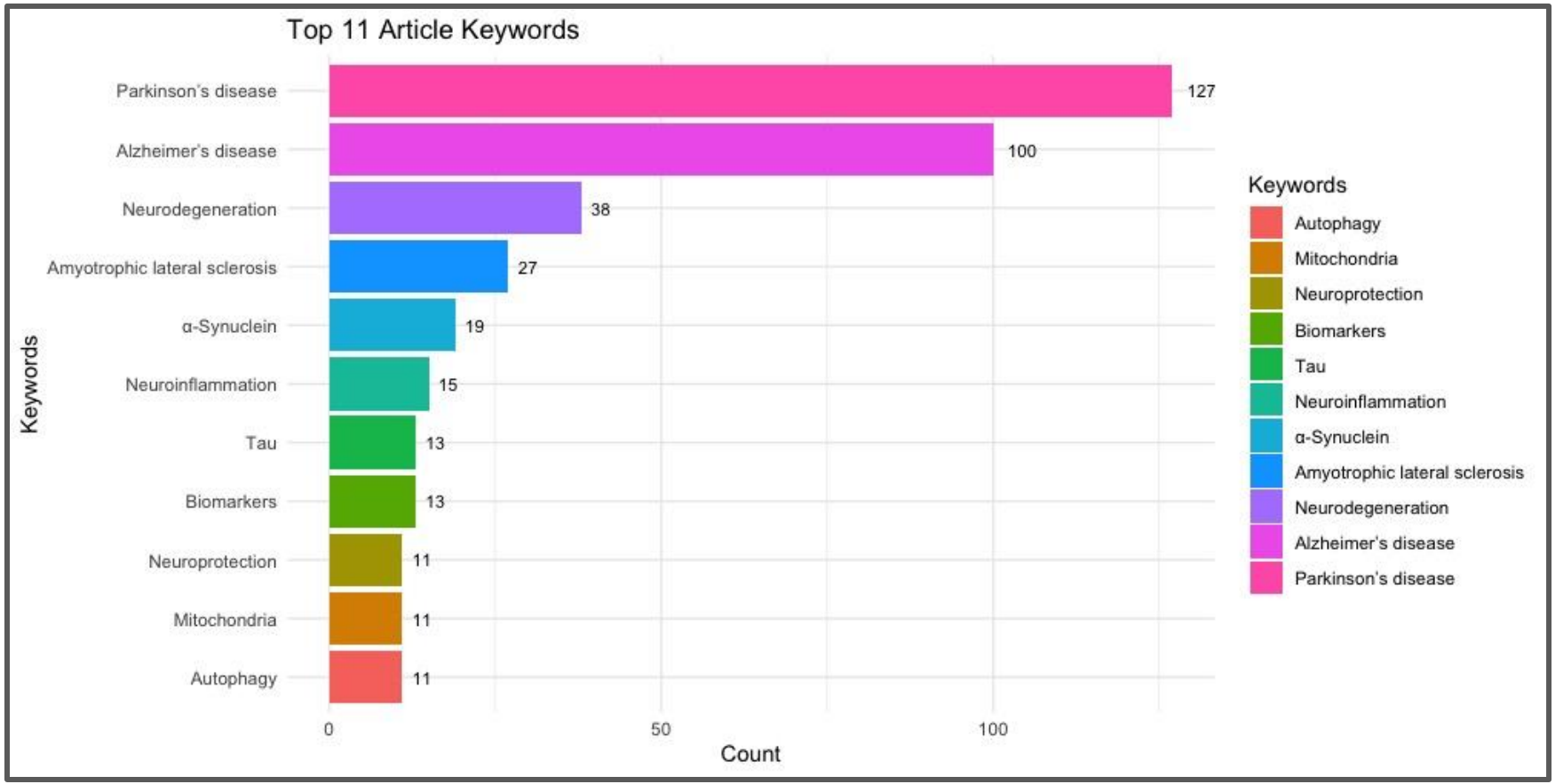
Some observations:

Duplicate records count: 1059
Non duplicate records count: 435
Non duplicate records without NA values: 403
Post preprocessing, records count: 403

Analysis and visualization:

```r
# Split the 'Keywords' column into a list of keywords and clean up whitespace
viz_project_dataframe <- preprocessed_articles_df %>% mutate(Keywords = str_squish(Keywords)) %>% tidyr::separate_rows(Keywords, sep = ",")
viz_project_dataframe <- viz_project_dataframe %>% mutate(Keywords = str_trim(Keywords, side = "left"))
# Count occurrences of each keyword
keyword_counts <- viz_project_dataframe %>% group_by(Keywords) %>% summarize(n = n())

# Filter for top 10 keywords
top_10 <- keyword_counts %>% top_n(10, n)

# Generate a palette of colors based on the number of unique keywords
top_10_colors <- scales::hue_pal()(length(unique(top_10$Keywords)))


# Reorder the Keywords based on their total counts
top_10$Keywords <- factor(top_10$Keywords, levels = top_10$Keywords[order(top_10$n)])

# Plot bar chart with numbers on top of bars and dynamic colors, with swapped axes
ggplot(top_10, aes(x = n, y = Keywords, fill = Keywords)) +
  geom_col() +
  geom_text(aes(label = n), hjust = -0.5, size = 3) +  # Add numbers on the bars
  labs(x = "Count") +
  ggtitle("Top 10 Article Keywords") +  # Add the chart title
  scale_fill_manual(values = top_10_colors) +  # Assign dynamic colors
  theme_minimal()  # Optional: Use a minimal theme for better readability
```

Bar Chart


Top 11 Article Keywords
Parkinson's disease
Alzheimer's disease
Neurodegeneration
Amyotrophic lateral sclerosis
α-Synuclein
Neuroinflammation
Tau
Biomarkers
Neuroprotection
Mitochondria
Autophagy
127
100
38
27
19
15
13
13
11
11
11
Keywords
0
50
100
Count
Keywords
Autophagy
Mitochondria
Neuroprotection
Biomarkers
Tau
Neuroinflammation
α-Synuclein
Amyotrophic lateral sclerosis
Neurodegeneration
Alzheimer's disease
Parkinson's disease

Challenges faced:

In scraping: Different abstract structure for different articles, hence made a conscious choice of picking the first paragraph, because usually abstract is a single paragraph.

While cleaning: Dropped articles with NA because couldn’t replace them with mean (there were no numerical values in data).

During analysis: Found duplicates in keywords, had to trim to remove spaces at the left of keywords.

Thank you!